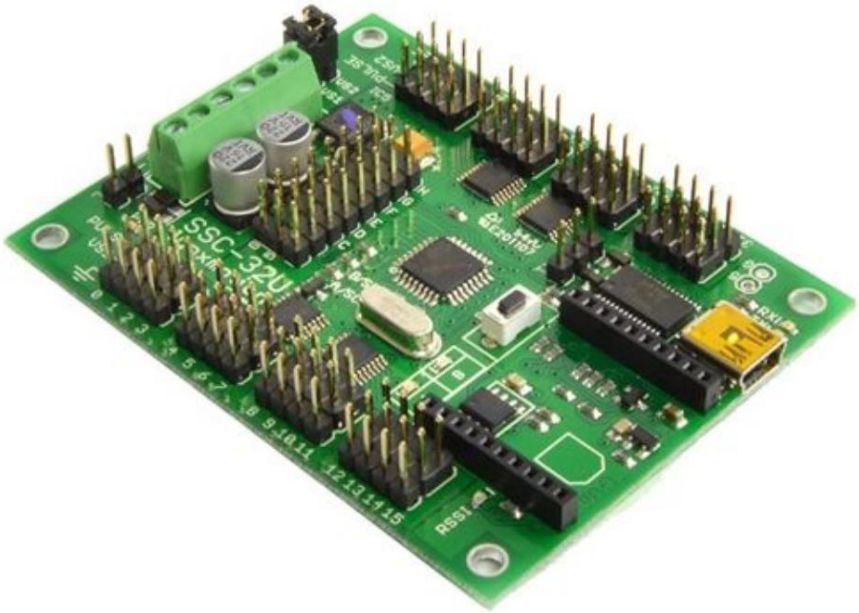
Imagine it. Build it. Control it.™

Carte contrôleur de servo USB Lynxmotion SSC-32U



Révisions

V1.2 Juillet 2017

Commentaires / Errata /
Assistance : http://www.robotshop.com/forum/ssc-32u-f142

Table des matières

Aperçu

Le Lynxmotion SSC-32U est un contrôleur de servomoteur R/C polyvalent et facile à utiliser, dont le cœur est un Atmel ATmega328p. Les fonctionnalités incluent : • Entrée USB, série ou XBee • Contrôle jusqu'à 32 servomoteurs • Séquenceur hexapode 12DoF intégré • 8 broches d'entrée analogique • Protocole de commande facile à comprendre • Fonctionnalités avancées • Idéal pour une utilisation avec des bras robotisés ou des robots à pattes • Compatible avec le logiciel graphique FlowBotics Studio

Notez que le SSC-32U est un contrôleur de servo dédié. En tant que tel, la carte n'est pas destinée à être « programmée » (ou à stocker du code créé par un utilisateur), mais plutôt à recevoir et à exécuter simplement des commandes qui lui sont envoyées par un périphérique externe tel qu'un ordinateur ou un microcontrôleur. L'utilisation d'un contrôleur de servo dédié libère de la mémoire sur un microcontrôleur qui serait utilisée pour mettre à jour en permanence les positions des servos.

Le SSC-32U est largement basé sur le SSC-32. Le guide d'utilisation du SSC-32 est disponible ici pour référence : http://www.lynxmotion.com/images/html/build136.htm

Servomoteurs numériques, analogiques et « intelligents »

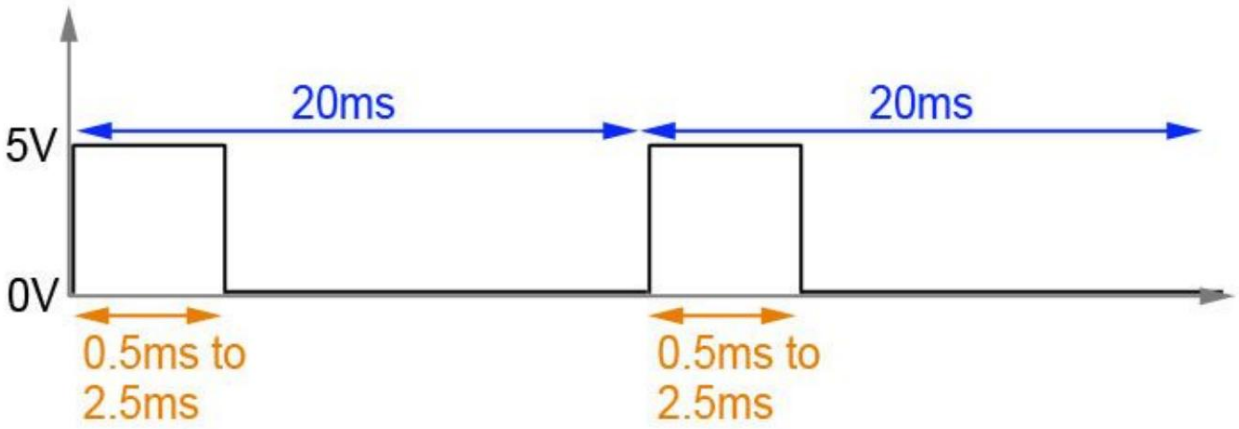
Servomoteurs analogiques

Notions de base sur les servos R/C

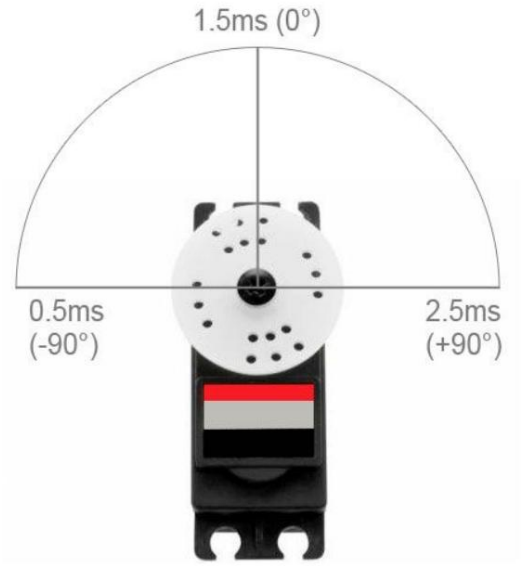
Lorsqu'il s'agit d'un servomoteur télécommandé (« R/C »), il est important de connaître la terminologie suivante : • Boîtier (normalement en plastique, parfois en aluminium) • Moteur à courant continu (normalement à balais, parfois sans balais ou plus exotique) • Engrenages (qui réduisent la vitesse du moteur et augmentent le couple) • Cannelure (connexion du dernier engrenage interne ; s'insère dans le palonnier du servo pour le faire tourner) • Palonnier (la pièce en plastique ou en métal qui peut être utilisée pour connecter d'autres éléments) • Potentiomètre (un capteur d'angle absolu utilisé pour fournir la position du servo) • Électronique (contrôle le mouvement) • Câble et connecteur à 3 broches (espacement standard de 0,1")

Blanc / jaune / orange = Signal (impulsion)
Rouge = 4,8 V à 6 V (VS)
Noir / marron = GND

Les servos R/C utilisent une forme de « modulation de largeur d'impulsion » (PWM). Dans ce cas, l'électronique du servo attend de recevoir un signal répété de 5 V qu'il chronomètre pour voir combien de temps ce signal reste activé. Pour la plupart des servos RC, une impulsion longue de 500 µs (0,5 ms) à 5 V ferait tourner le palonnier du servo à -90 degrés ; une impulsion longue de 2 500 µs (2,5 ms) à 5 V ferait tourner le palonnier à +90 degrés ; par conséquent, une impulsion de 1 500 µs correspond à 0 degré (centré). Cette impulsion doit être répétée toutes les ~20 ms.



Impulsion temporisée R/C (répétée toutes les 20 ms)



Servomoteur standard Hitec montrant l'angle de fonctionnement par rapport au signal

Lorsqu'un servo RC atteint un angle spécifique, il fera de son mieux pour conserver cette position, à condition de continuer à recevoir le signal toutes les 20 ms. Si le couple est supérieur à ce que le servo peut fournir, le courant augmentera considérablement, ce qui entraînera un échauffement du servo et éventuellement une panne.

La plupart des servos R/C ont un cycle de service d'environ 25 %. Le « cycle de service » correspond à la durée pendant laquelle le moteur doit être allumé ou éteint. Cela ne signifie pas que vous pouvez faire fonctionner un servomoteur pendant 1 heure et le laisser refroidir pendant 3 heures ; nous suggérons qu'un servomoteur normal soit en fonctionnement constant pendant moins de 10 minutes environ, après quoi il aura idéalement besoin d'environ 30 minutes pour refroidir.

Servomoteur à rotation continue

Un servomoteur RC à rotation continue utilise les mêmes impulsions/commandes qu'un servomoteur RC analogique normal, mais convertit les impulsions en vitesse et en direction, où 1,5 ms correspond à l'arrêt, 0,5 ms à la pleine vitesse dans le sens inverse des aiguilles d'une montre et 2,5 ms à la pleine vitesse dans le sens des aiguilles d'une montre. Les valeurs intermédiaires correspondent aux vitesses intermédiaires. Un servomoteur à rotation continue ne peut pas être affecté à un angle spécifique (ou relatif) et fonctionne un peu comme un moteur à engrenages à courant continu.

Servo numérique R/C

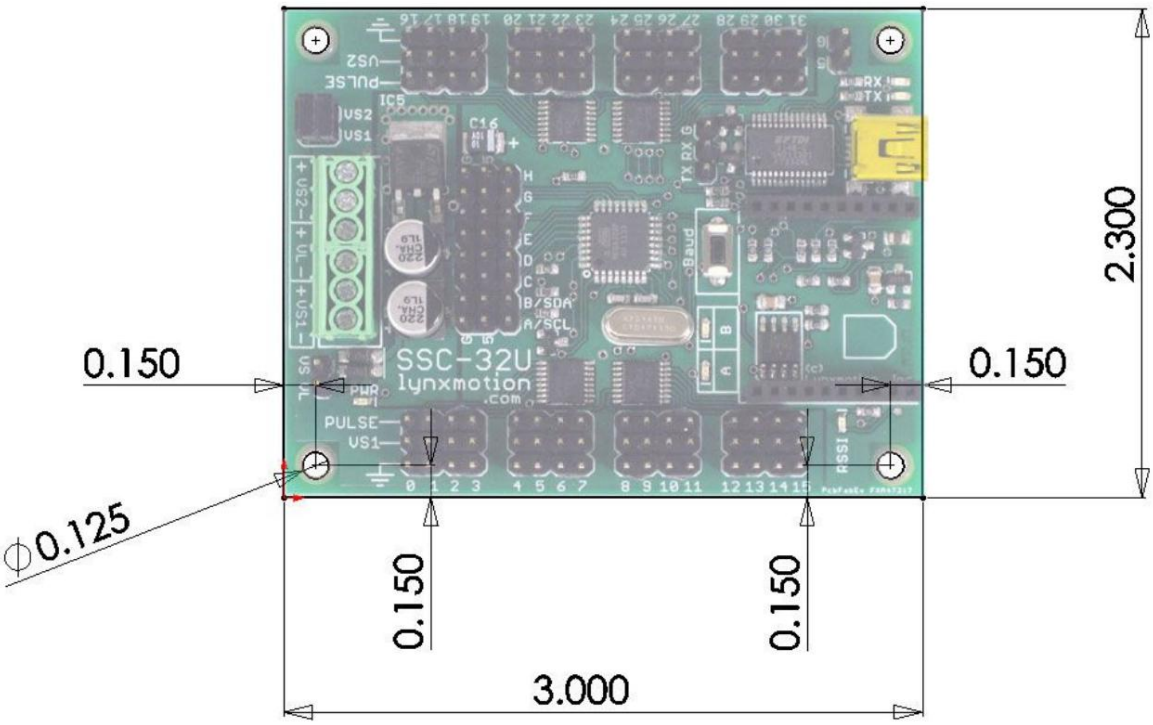
Un servomoteur numérique fonctionne de la même manière qu'un servomoteur analogique R/C normal, mais permet souvent des mouvements plus précis/plus rapides. Dans de nombreux cas, les limites et la position centrale du servomoteur (ainsi qu'une variété d'autres paramètres) peuvent être modifiées à l'aide d'un programmateur de servomoteur (le SSC-32U n'est pas un programmateur de servomoteur numérique). Une fois le servomoteur programmé, il peut être utilisé comme n'importe quel autre servomoteur analogique R/C. Les servomoteurs numériques ressemblent aux servomoteurs analogiques.

Servo intelligent

Un « Smart Servo » est un servo qui n'est normalement pas contrôlé par un signal R/C mais plutôt par des commandes série. Le connecteur d'un servo intelligent est normalement différent de celui d'un servo R/C.
Le SSC-32U n'est pas conçu pour contrôler les servomoteurs intelligents. Les servomoteurs intelligents ont souvent conceptions mécaniques différentes / personnalisées de celles des servos R/C.

Informations sur le matériel

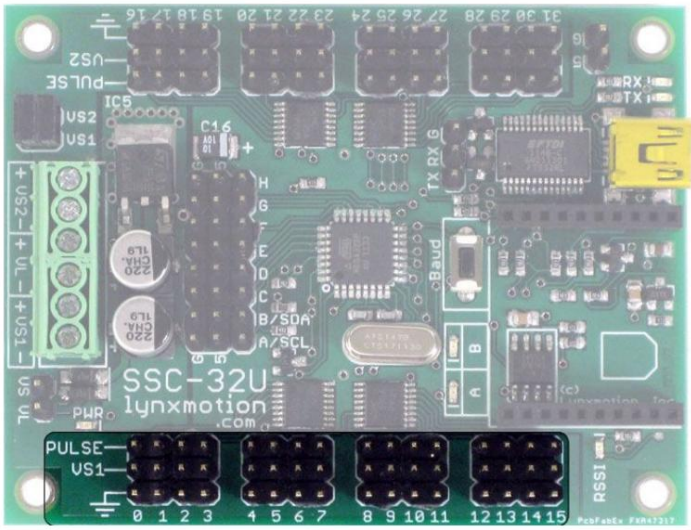
Dimensions



Dimensions de la carte SSC-32U (pouces)

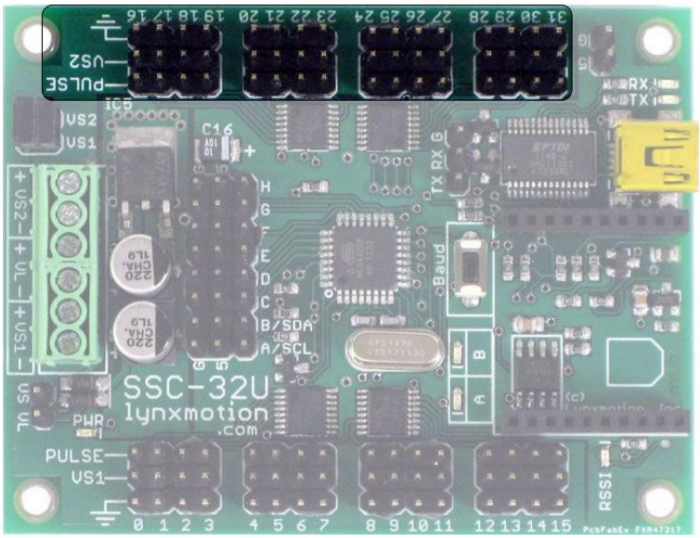
Le SSC-32U mesure 3,00" x 2,30" avec des trous de 0,125" placés à 0,15" de chaque bord. Il a été conçu aux mêmes dimensions que le Lynxmotion BotBoarduino, le Lynxmotion SSC-32 (son prédécesseur), le Lynxmotion Bot Board et le Bot Board 2 afin qu'ils puissent être empilés.

Broches S1



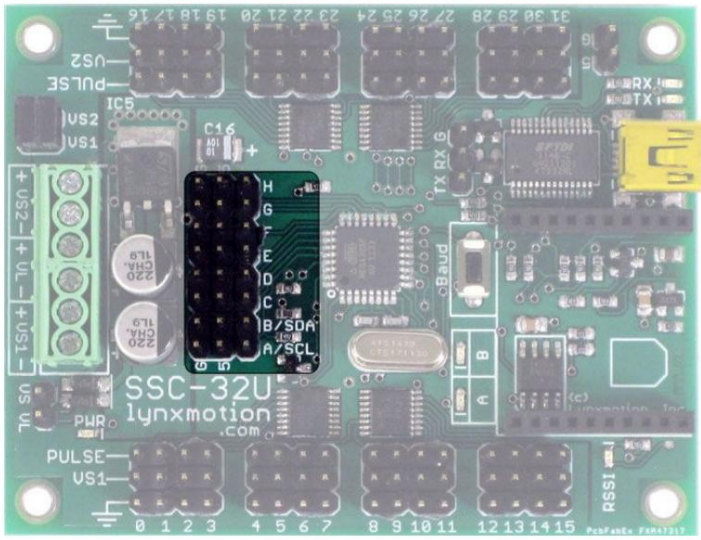
Le contrôleur de servo SSC-32U peut contrôler jusqu'à 32 servos, qui sont disposés en deux sections distinctes de 16 servos, chacune regroupée en groupes de quatre. Les broches 0 à 15 correspondent à VS1.
La broche la plus à l'extérieur (la plus éloignée du centre de la carte) est la broche de masse (normalement le fil noir dans un câble à trois broches). La broche centrale correspond à la tension (normalement le fil rouge) et est marquée VS1. La dernière broche est la broche de signal qui envoie la commande de position au servo. La couleur du fil de signal peut varier, mais elle est normalement blanche ou jaune.

Broches S2



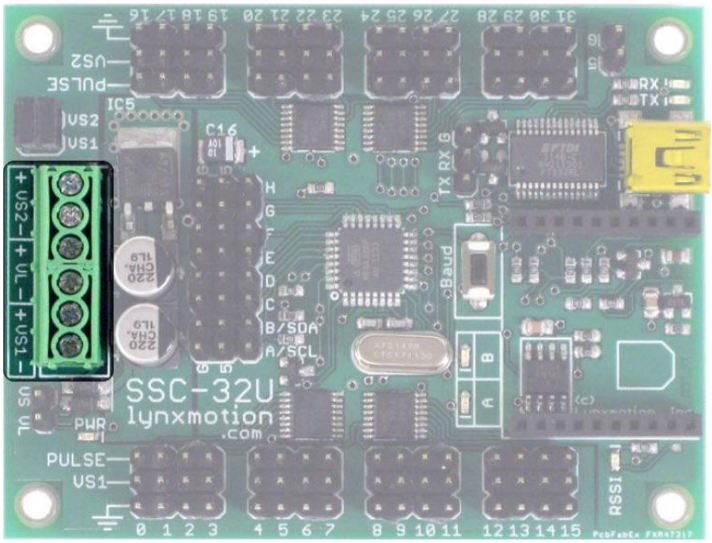
Il s'agit du deuxième ensemble de 16 broches. Les broches 16 à 31 correspondent à VS2. La broche la plus à l'extérieur (la plus éloignée du centre de la carte) est la broche de masse (normalement le fil noir dans un câble à trois broches). La broche centrale correspond à la tension (normalement le fil rouge) et est marquée VS1. La dernière broche est la broche de signal qui envoie la commande de position au servo. La couleur du fil de signal peut varier, mais elle est normalement blanche ou jaune.

Broches d'E/S analogiques



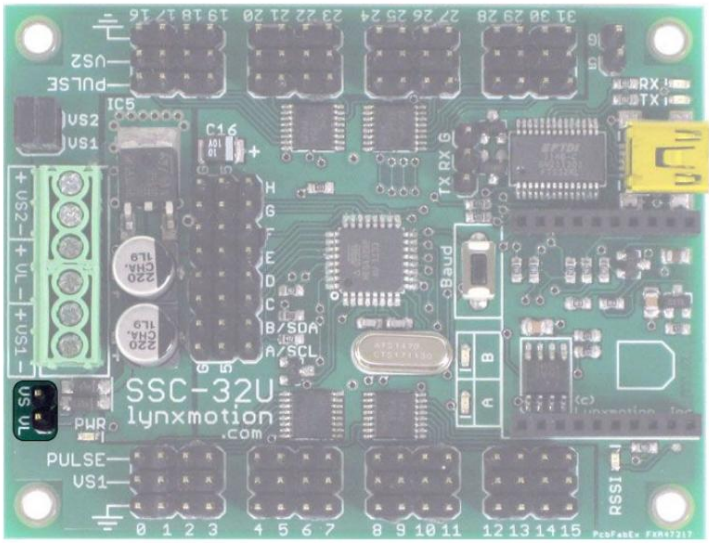
La carte comporte 8 broches d'E/S, étiquetées de « A » à « H ». La rangée de broches de gauche marquée « G » correspond à la masse (normalement le fil noir sur un câble à trois broches). La rangée centrale de broches, marquée « 5 », fournit une sortie de 5 V (normalement le fil rouge). La dernière rangée de broches (sans étiquette) est destinée au signal. Deux de ces broches peuvent également être utilisées pour I2C, la broche A correspondant à SCL et la broche B correspondant à SDA. Pour la communication I2C, vous devez également connecter la broche GND, et si le SSC-32 alimente les autres périphériques I2C, vous devez également connecter l'une des broches 5 V.

Bornes à vis



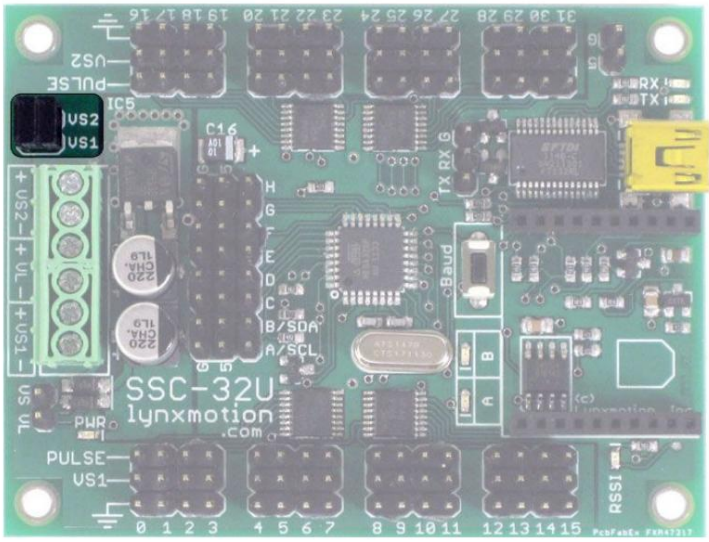
VS1 fournit l'alimentation directement aux broches 0 à 15 (idéalement 4,8 V à 6 V pour les servos RC standard)
VS2 fournit l'alimentation directement aux broches 16 à 31 (idéalement 4,8 V à 6 V pour les servos RC standard)
VL fournit l'alimentation au contrôleur logique. Assurez-
vous de connecter le fil positif (généralement rouge) au « + » et le fil négatif (généralement noir) au « - ». Assurez-vous qu'aucun fil détaché n'entre en contact avec les deux en même temps (provoquant un court-circuit).
Veuillez vous référer à la section Alimentation de ce guide pour obtenir des informations importantes.

VS = VL



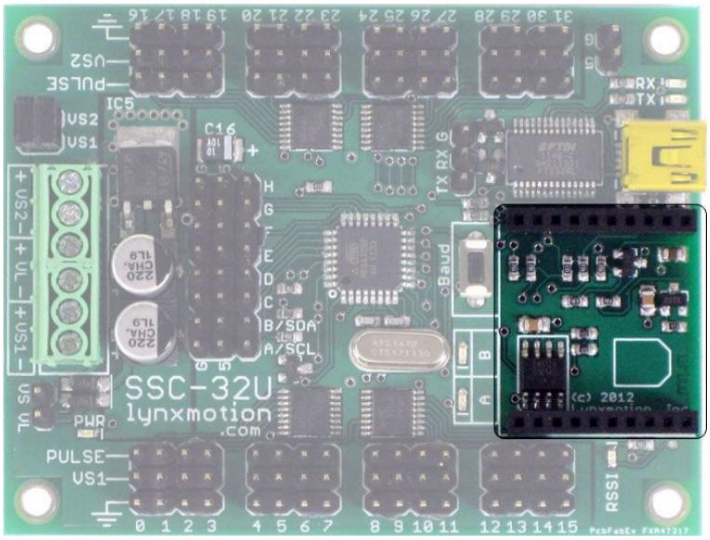
Ces deux broches permettent de relier la borne VS1 à VL (c'est-à-dire VL = VS). Cela aurait le même effet que de faire passer un fil de la borne à vis VS1+ à la borne à vis VL+.
Veuillez vous référer à la section Alimentation de ce guide pour obtenir des informations importantes. Aucun cavalier n'est fourni pour ces broches.

VS1 = VS2



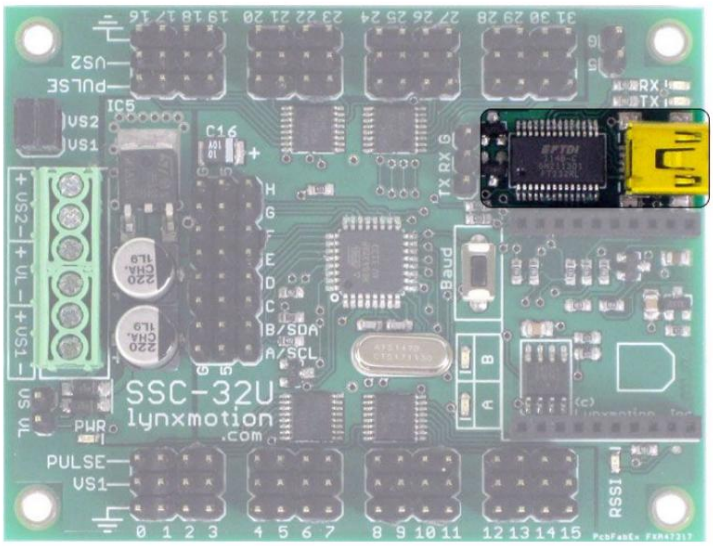
Ces broches 2x2 vous permettent de définir VS1 = VS2. Cela aurait le même effet que de faire passer un fil de la borne à vis VS1+ à la borne à vis VS2+. Il existe deux cavaliers pour gérer un courant plus élevé. La carte est livrée avec ceux-ci en place et il appartient à l'utilisateur de les retirer si nécessaire. Veuillez vous référer à la section Alimentation de ce guide pour obtenir des informations importantes.

XBee



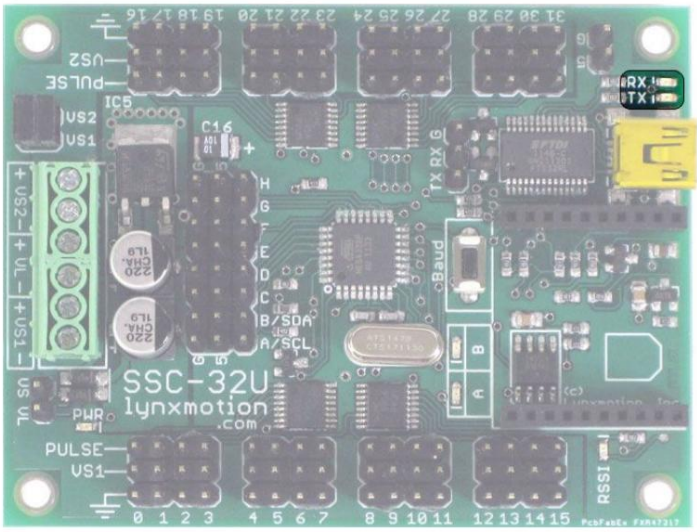
Les connecteurs XBee vous permettent de connecter un module sans fil XBee à d'autres appareils compatibles XBee. Prenez note de l'image sur la carte pour savoir comment orienter le module. Veuillez vous référer à la section communication/contrôle de ce guide pour plus d'informations.

USB



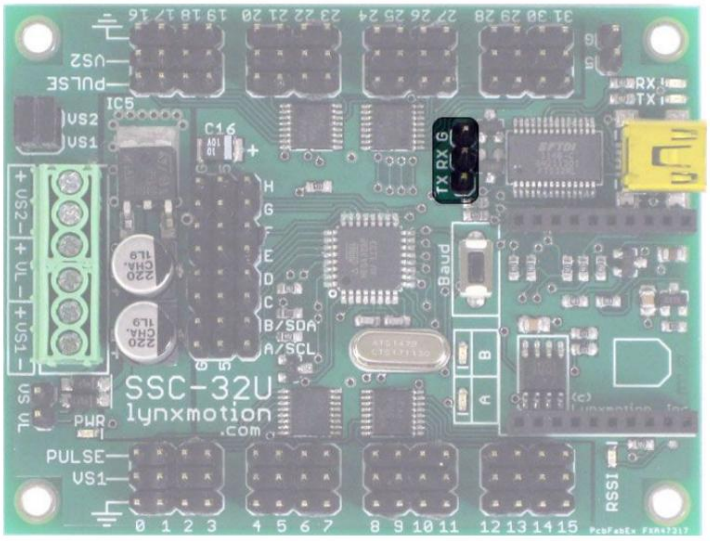
Les commandes peuvent être envoyées au microprocesseur du SSC-32U depuis l'ordinateur via le port USB intégré. Le microprocesseur ne peut comprendre que les commandes envoyées au format série (et non USB), donc une puce FTDI (la zone en surbrillance à côté du port USB) convertit le flux de données USB en série et permet à la carte d'être récupérée par le port COM de votre ordinateur une fois connectée.
Veuillez vous référer à la section pilotes ci-dessous pour plus d'informations.

Tx-Rx



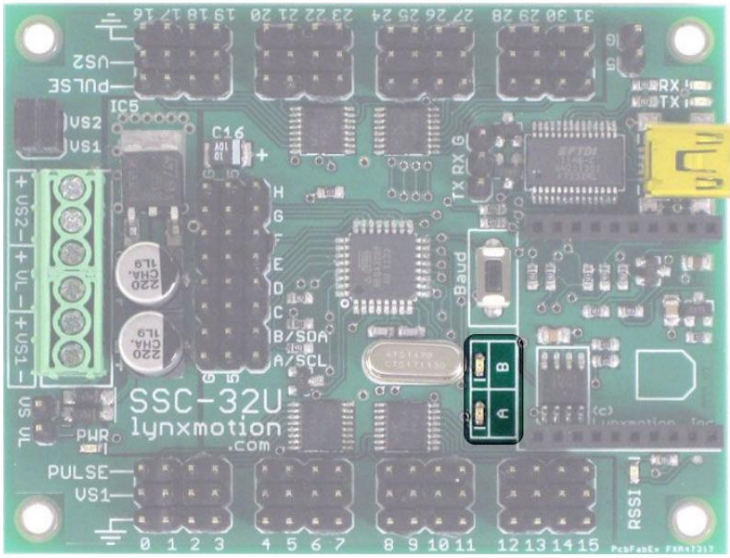
Les deux LED situées à côté du port USB fournissent à l'utilisateur un retour visuel sur la réception (Rx) et la transmission (Tx) des données. Ces informations sont uniquement associées à la communication via le port USB et non à l'en-tête XBee ou aux broches série.

En série



Les trois broches situées derrière la puce FTDI sont Tx, Rx et GND. Les commandes envoyées depuis le SSC-32U sont effectuées via la broche Tx tandis que les commandes à recevoir par le SSC-32U sont effectuées via la broche Rx. Ces broches vous permettent d'envoyer facilement des commandes au contrôleur de servo depuis un autre microcontrôleur. Pour ce faire, connectez la broche Tx du microcontrôleur à la broche Rx du SSC-32U, la broche Rx du microcontrôleur à la broche Tx du SSC-32 et GND à GND.

LED A/B



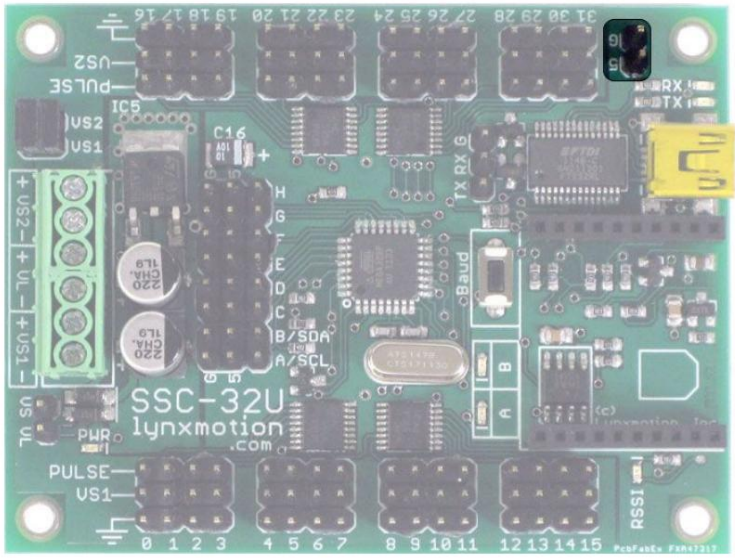
Il y a deux LED distinctes sur la carte étiquetées A et B. Lorsque vous ne définissez pas le débit en bauds, le

Les LED indiquent les éléments suivants :

- Les deux sont allumés à la mise sous tension avant la réception d'un octet. • Le vert clignote lorsqu'un octet valide est reçu. • Le rouge clignote lorsqu'un octet non valide est reçu (erreur de cadrage).

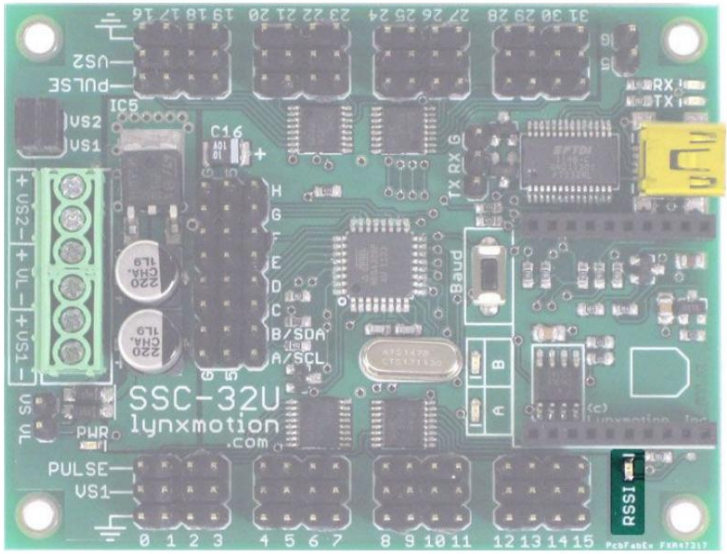
Pour plus d'informations, veuillez vous référer à la section Baud de ce guide.

Broches 5G



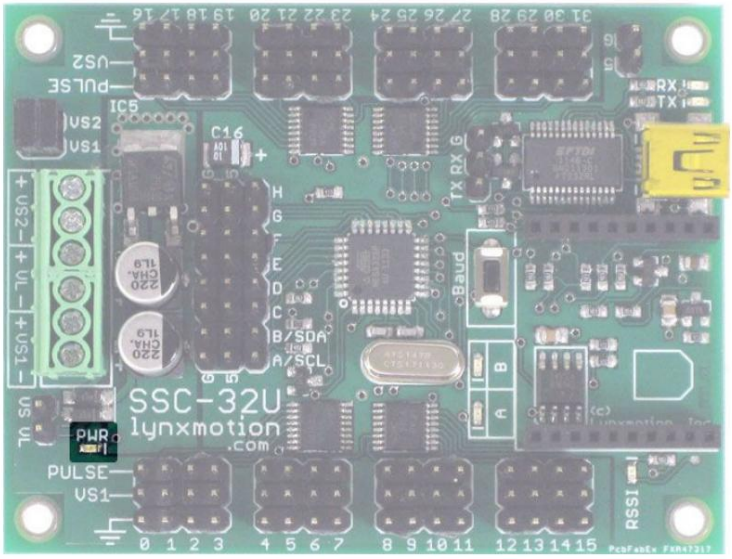
Ces broches ne sont normalement pas nécessaires et ne sont pas fournies. Si les deux condensateurs intégrés ne suffisent pas pour votre application, un troisième condensateur peut être ajouté ici. Si vous n'êtes pas très versé en électronique, nous vous suggérons de ne pas toucher à cette partie de la carte.

RSSI



RSSI signifie « Received Signal Strength Indication » (Indication de la force du signal reçu) et la LED est connectée à la broche 6 de la prise Xbee. Différents intervalles de clignotement sont utilisés pour indiquer différents états en fonction du module choisi.

Réducteur à eau sous pression



La LED PWR est connectée à V_logic qui est l'équivalent de MAX(VL, VS1) moins environ 0,7 V. C'est la tension qui est ensuite régulée à 5 V pour alimenter les circuits logiques. La LED s'allume lorsque le microcontrôleur embarqué reçoit une alimentation adéquate.

Options d'alimentation

USB

Lorsque la carte est connectée à un ordinateur via USB, la puce USB vers série intégrée sera alimentée et votre ordinateur pourra la détecter et installer les pilotes. L'USB n'alimente cependant pas la puce ATmega principale, elle doit donc être alimentée séparément via VS1 ou VL.

Borne à vis VL

Les bornes à vis VL permettent des entrées non régulées vers la tension logique (la tension utilisée par le processeur principal/la puce). La tension logique est automatiquement sélectionnée entre VL et VS1 (celle qui a la tension la plus élevée). Tant que VS1 est supérieur à 5,3 V (sans compter les chutes de tension temporaires), il suffit d'alimenter la tension logique et vous n'avez pas besoin de connecter quoi que ce soit aux bornes VL.

- Idéal : Rien de connecté • Nominal : 6-12 V
- Absolu : 5,3~16 V

Dans la plupart des situations, une seule batterie est nécessaire (connectée à VS1), bien qu'il puisse y avoir des situations où il est préférable d'avoir une batterie pour la logique et une autre pour les servos : • Vous utilisez 4,8 V pour votre VS1, ce qui n'est pas suffisant pour VL. • Vous souhaitez couper l'alimentation des moteurs mais garder la logique sous tension. • Vous souhaitez vous assurer que la logique est toujours alimentée même si la batterie du servo est épuisée.

Borne à vis VS1

La borne VS1 se connecte directement aux lignes d'alimentation et de masse des broches de servomoteur 0 à 15. La tension appliquée à VS1 doit idéalement correspondre à la tension nominale des servomoteurs. Pour la plupart des servomoteurs R/C, elle est de 6 V. Certains servomoteurs R/C « haute tension » peuvent fonctionner avec une batterie LiPo de 7,4 V ou 11,1 V, mais à moins que cela ne soit clairement indiqué dans les spécifications du servomoteur, faites attention à la tension que vous utilisez. L'utilisation d'une batterie LiPo de 7,4 V avec un servomoteur R/C normal est déconseillée et l'utilisation d'une batterie de 11,1 V avec un servomoteur normal le détruira probablement.

- Nominal : 6 V (servos R/C standard) • Absolu : 0~16 V

Borne à vis VS2

La borne VS2 est connectée directement aux lignes d'alimentation et de masse des broches 16 à 31 du servomoteur. La tension appliquée à VS2 doit idéalement correspondre à la tension nominale des servomoteurs. Pour la plupart des servomoteurs R/C, elle est de 6 V. Certains servomoteurs R/C « haute tension » peuvent fonctionner avec une batterie LiPo de 7,4 V ou 11,1 V, mais à moins que cela ne soit clairement indiqué, faites attention à la tension que vous utilisez.

Cavaliers VS1 = VS2 La
seconde moitié des broches du servo numérotées de 16 à 31 peut être alimentée à l'aide de la même source d'alimentation que VS1 en laissant les deux cavaliers VS1 = VS2 en place. Il y a deux cavaliers (au lieu d'un seul) en raison du courant impliqué. Si vous souhaitez alimenter la ligne de servos connectée à VS2 séparément de VS1, vous pouvez retirer ces deux cavaliers. Si vous avez les deux bornes VS1 = VS2 en place, vous pouvez alimenter SOIT VS1 soit VS2, mais pas les deux. Voici quelques exemples de cas où le retrait de ces cavaliers est bénéfique :

- Une deuxième batterie est nécessaire en raison du courant élevé
- Le deuxième jeu de servos fonctionne à une tension différente
- Vous souhaitez utiliser deux packs séparés pour l'alimentation

VL = VS

Le cavalier VL=VS est un moyen de forcer la tension logique à utiliser l'alimentation connectée à VS1.
À moins que vous ne rencontriez des problèmes inhabituels avec la sélection automatique de l'alimentation, ce cavalier ne doit pas être utilisé. Ce cavalier n'est pas installé ni inclus dans le package. N'utilisez pas les bornes à vis VL si vous installez ce cavalier.

Considérations relatives à l'alimentation

La carte SSC-32U ne lit pas la tension de la batterie et continue de fonctionner même si la batterie est en cours de décharge profonde. Il appartient à l'utilisateur de veiller à ce que la tension de la batterie soit maintenue au-dessus d'une certaine tension, sinon il y aura un risque d'endommager irrémédiablement la batterie.

Les symptômes indiquant qu'une batterie est déchargée sont les suivants : • La vitesse du servo est inférieure à la normale • Les servos ne peuvent pas maintenir leur position sous charge

Adaptateur mural

Si vous souhaitez alimenter la carte à l'aide d'un adaptateur mural plutôt que de piles, les éléments suivants doivent être pris en considération :

- Un adaptateur mural peut fournir une alimentation infinie, bien que les servos ne doivent pas fonctionner en continu pendant plus de 30 minutes environ. • L'adaptateur mural doit pouvoir fournir suffisamment de courant (évalué en ampères) pour tous les servos. • La polarité du connecteur cylindrique n'est pas toujours évidente ; assurez-vous de connecter le positif au positif et le négatif au négatif / GND. • Étant donné que le SSC-32U n'a pas de connecteur cylindrique, un adaptateur est probablement nécessaire. Nous suggérons un adaptateur de prise cylindrique de 2,1 mm vers borne à vis et une paire de fils, ou un faisceau de câbles approprié.

Options d'entrée du connecteur cylindrique

Chimie et courant des batteries

Alcalin

Les piles alcalines sont généralement vendues sous forme de piles simples de 9 V, AAA, AA, C et D. Ces piles ne sont pas rechargeables et sont généralement de 1,5 V. Vous pouvez utiliser quatre piles de 1,5 V pour fabriquer un pack de piles de 6 V. La capacité de la pile est rarement indiquée sur les piles, il est donc difficile d'évaluer exactement leur durée de vie. Si la pile est bon marché, vous pouvez supposer qu'elle ne durera pas longtemps. Le taux de décharge (la quantité de courant que la pile peut fournir en rafales ou en continu) est également rarement indiqué. Par conséquent, si vous utilisez plusieurs servos et que vous constatez qu'ils n'ont pas assez de puissance, vos piles en sont probablement la cause.

NiMh / NiCd

Les piles NiMh et NiCd sont généralement disponibles sous forme de piles AA, AAA ou de packs de batteries. Elles peuvent être rechargées et sont actuellement le type de batterie le plus populaire pour les robots mobiles (à part les humanoïdes) en raison de leur poids et de leur prix. Les piles NiMh sont préférables aux piles NiCd car elles peuvent être rechargées à pleine capacité à chaque fois, alors que si vous rechargez des piles NiCd qui ne sont que partiellement utilisées, elles perdront une partie de leur capacité à chaque fois. Les piles à cellule unique fournissent généralement 1,2 V et, par conséquent, pour fabriquer un pack de batteries de 6 V, vous auriez besoin de 5 cellules. La capacité de chaque cellule est presque toujours indiquée sur la cellule. Le taux de décharge tend à être de 1C ou 2C pour la plupart des batteries à base de nickel. La valeur « C » est en relation avec la capacité. Si une cellule est évaluée à 2800 mAh, 1C = 2800 mA. Par exemple, une cellule NiMh de 1,2 V et 2000 mAh évaluée à 1C peut se décharger à 2,0 A en continu. En utilisant cinq de ces batteries, vous pourrez fournir 6 V à 2 A (en continu). Les fournisseurs/fabricants de batteries créent des packs de plusieurs cellules pour fournir des tensions plus élevées et ajoutent un connecteur personnalisé.

Lithium

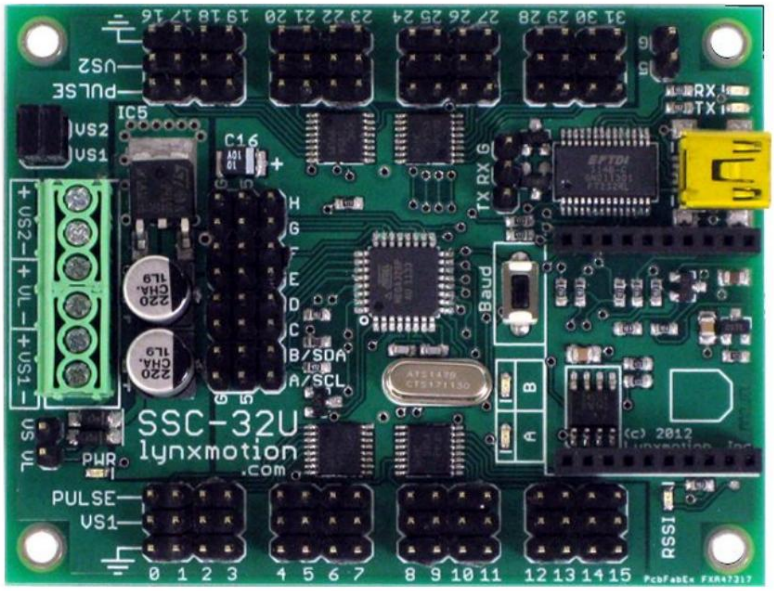
Les batteries LiPo / LiFe, etc. se présentent généralement sous forme de cellules plates ou de packs rectangulaires et se présentent normalement sous forme de multiples de 3,7 V. Les batteries au lithium ont l'avantage d'être plus légères que les batteries NiMh et capables de fournir des taux de décharge nettement plus élevés (10 C, 20 C ou même plus). Malheureusement, pour les servos R/C, 3,7 V est trop faible, alors que 7,4 V est (généralement) trop élevé. Nous suggérons d'utiliser des batteries LiPo uniquement pour les servos haute tension.

Acide de plomb

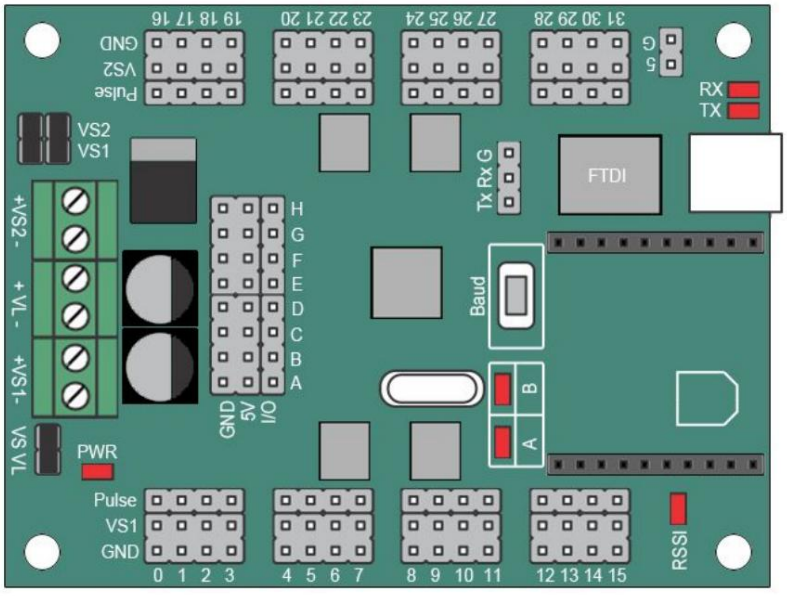
Les batteries au plomb sont généralement de 6 V, 12 V et 24 V, bien que de nombreuses autres options soient possibles. C'est le type de batterie que l'on trouve normalement dans un véhicule à essence. Une batterie au plomb de 6 V peut normalement fournir suffisamment de courant pour alimenter de nombreux servos R/C, mais la batterie elle-même sera assez lourde.

Options de communication/contrôle Il existe trois façons de
communiquer avec la carte SSC-32U : USB, TTL UART et socket XBee. Nous vous recommandons d'en utiliser un seul, mais l'USB aura la priorité si plusieurs sont utilisés.

Le dessin ci-dessous représentera le SSC-32 et sera utilisé pour montrer les connexions entre le SSC-32U et autres appareils :



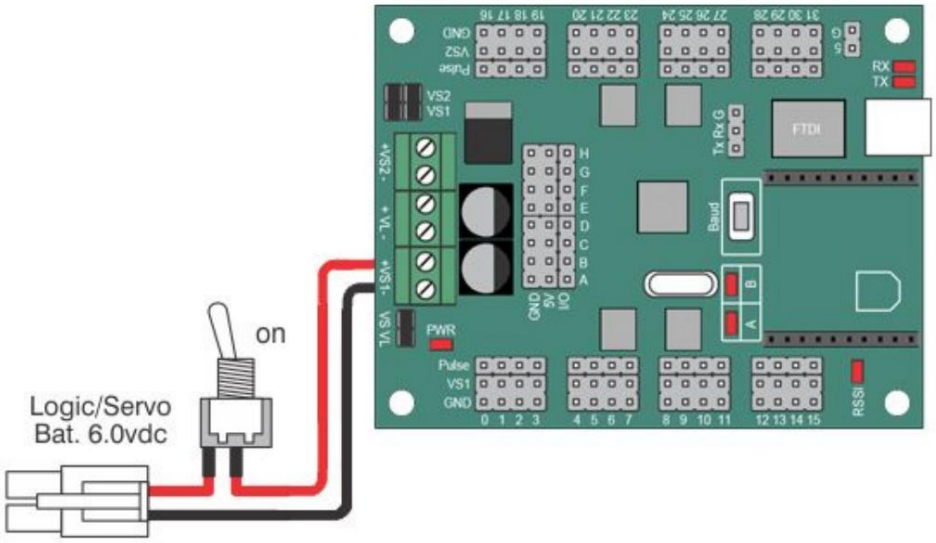
SSC-32U



Dessin du SSC-32U

USB

Vous pouvez connecter la carte à un ordinateur de bureau ou portable standard à l'aide d'un câble USB normal (miniB vers A). Une fois le câble USB connecté, votre ordinateur devrait le détecter automatiquement et installer les pilotes appropriés pour que la carte apparaisse comme un port COM et autoriser les programmes de votre ordinateur à communiquer avec la carte. Si votre ordinateur ne détecte pas la carte, vous devez télécharger et installer manuellement les pilotes VCP de la puce FTDI (en fonction de votre système d'exploitation) : http://www.ftdichip.com/Drivers/VCP.htm



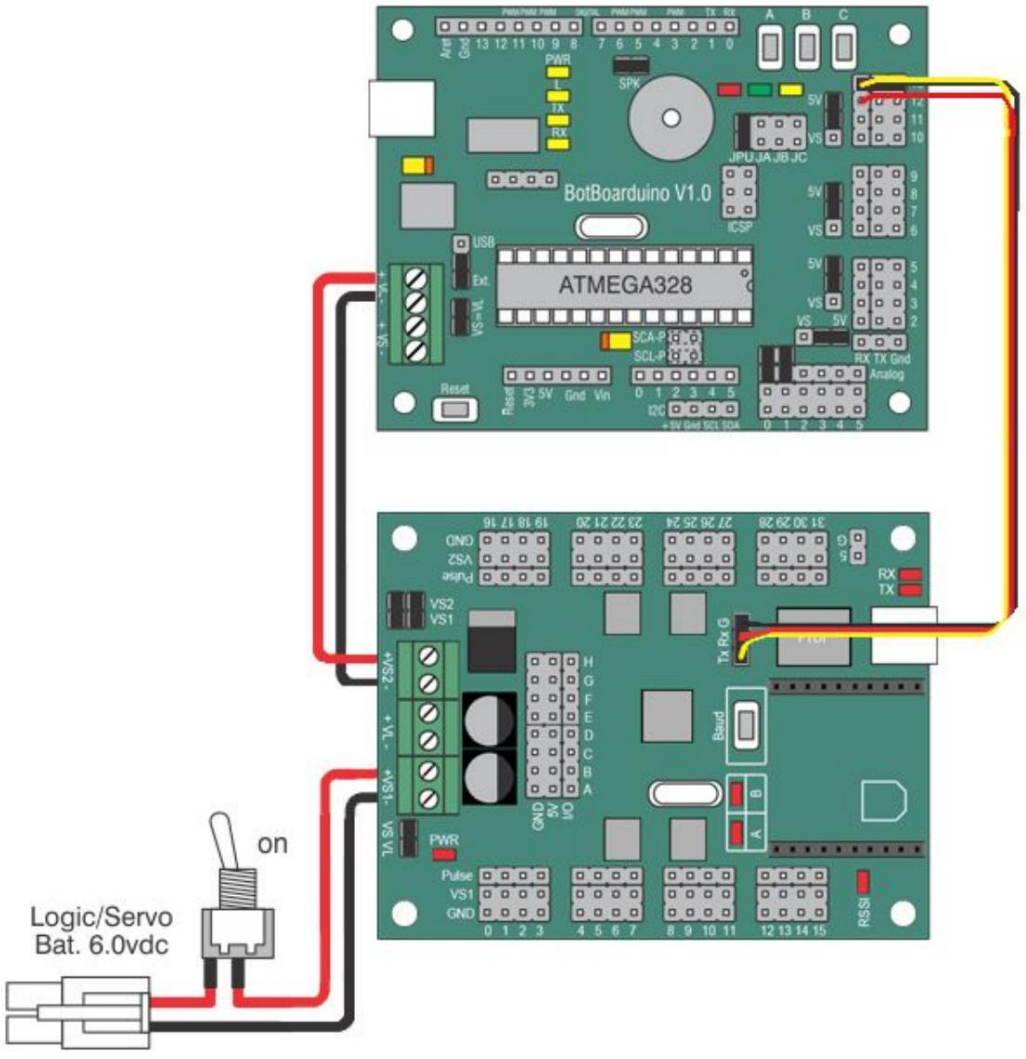
Alimentation SSC-32U

Notez que tous les câbles USB ne sont pas identiques et que certains fournissent uniquement de l'énergie plutôt que de l'énergie et de la communication. Si vous rencontrez des problèmes de communication via USB avec la carte, vérifiez d'abord l'alimentation, puis changez le câble USB.

TTL UART (broches TX / RX / G)

Les commandes peuvent être envoyées au SSC-32 à l'aide des broches série de la carte. Le BotBoarduino peut être connecté au SSC-32 de deux manières

1. Connexion série directe
   - TX du BotBoarduino vers RX sur le SSC-32U
   - RX du BotBoarduino vers TX sur le SSC-32U
   - GND sur le BotBoarduino vers GND sur le SSC-32U
2. Logiciel série (vous permettant de connecter d'autres appareils aux broches série du BotBoarduino).
   - Broche E/S 13 du BotBoarduino vers SSC-32 TX (jaune) • Broche E/S 12 du BotBoarduino vers SSC-32 RX (rouge) • GND du BotBoarduino vers GND du SSC-32 (noir)

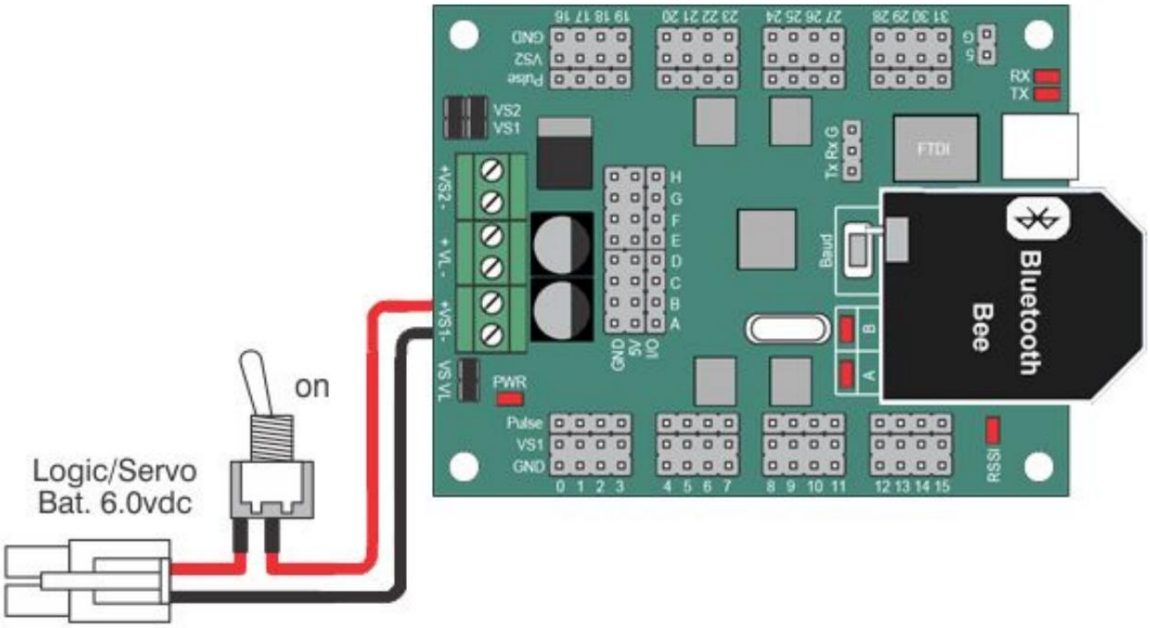


SSC-32U vers BotBoarduino / Arduino

En-têtes XBee

La prise XBee du SSC-32U peut être utilisée avec une variété d'appareils sans fil, notamment : • Modules Bluetooth avec empreinte XBee • Modules XBee

• Modules RF avec empreinte XBee

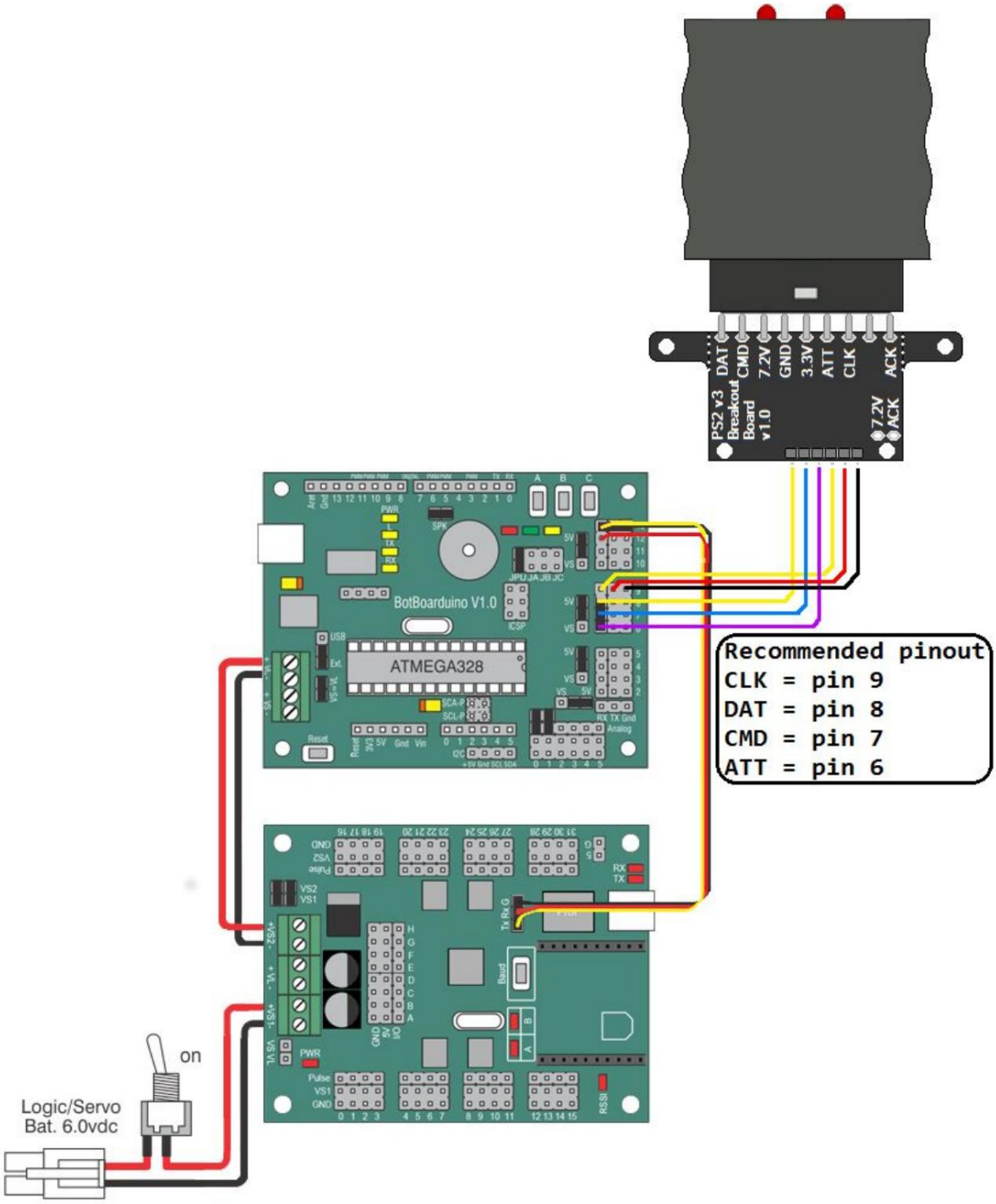


SSC-32U avec Bluetooth Bee installé

Notez que l'orientation du module correspond à celle de la sérigraphie sur la carte.

SSC-32U / BotBoarduino / PS2

Le SSC-32U peut être utilisé avec le BotBoarduino (et les Arduinos normaux), qui est lui-même éventuellement connecté à un contrôleur PS2. Le récepteur PS2 et le levelshifter de Lynxmotion sont présentés.



SSC-32U avec BotBoarduino et récepteur Lynxmotion PS2 V3 + Level Shifter

Commandes

Types et groupes de commandes.

Commandes de servo unique

Pour que le SSC-32 puisse positionner un servo, il doit recevoir une commande série au format suivant. Notez que les signes inférieur et supérieur ne sont pas nécessaires. Les valeurs en italique sont facultatives.

<ch> P <pw> S <spd> T <heure> <cr>

- <ch> : broche / canal auquel le servo est connecté (0 à 31) en décimal • <pw> : largeur d'impulsion souhaitée (normalement 500 à 2500) en microsecondes • <spd> : vitesse de déplacement du servo en microsecondes par seconde* • <time> : temps en microsecondes pour passer de la position actuelle à la position souhaitée. Cela affecte tous les servos (65535 max) *
- <cr> : retour chariot (ASCII 13)**

Exemple 1 : #5P1500S750<cr>

Cela permettrait au servomoteur connecté à la broche n°5 du SSC-32U de se déplacer vers la position 1500 (0 degré / centré) à une vitesse de 750 uS par seconde*. Les arguments numériques de toutes les commandes SSC-32 doivent être des chaînes ASCII de nombres décimaux, par exemple « 1234 ». Certaines commandes acceptent des nombres négatifs, par exemple « -5678 ». Le format ASCII n'est pas sensible à la casse. Utilisez autant d'octets que nécessaire. Les espaces, les tabulations et les sauts de ligne sont ignorés.

*Pour mieux comprendre l'argument de la vitesse, considérez que 1000 uS de déplacement entraîneront environ 90° de rotation. Une valeur de vitesse de 100 uS par seconde signifie que le servo mettra 10 secondes (diviser 1000 par 100) pour se déplacer de 90°. Alternativement, une valeur de vitesse de 2000 uS par seconde équivaut à 500 mS (une demi-seconde) pour se déplacer de 90° (diviser 1000 par 2000). Notez que les servos ont une vitesse maximale (en degrés par seconde), donc même si vous pouvez essayer d'assigner une vitesse plus rapide, le servo sera toujours physiquement limité.

**Toutes les commandes SSC-32 doivent se terminer par un caractère de retour chariot (ASCII 13). Dans Arduino, cela peut être fait en utilisant la commande Serial.println(); . Plusieurs commandes du même type peuvent être émises simultanément dans un groupe de commandes. Toutes les commandes d'un groupe de commandes seront exécutées après la réception du retour chariot final.

Exemple 2 : #3 P1600 T1000 <cr>

L'exemple 2 déplacera le servo 3 vers la position 1600. Il faudra 1 seconde pour terminer le déplacement, quelle que soit la distance que le servo doit parcourir pour atteindre la destination.

Commande de servo multiple (également appelée « groupe de commandes »)

Le SSC-32U permet de recevoir plusieurs commandes dans la même chaîne. Pour la position du servo, cela peut ressembler à ceci :

<ch> P <pw> S <vit> ... # <ch> P <pw> S < vit> T <heure> <cr>

Les commandes de types différents ne peuvent pas être mélangées dans le même groupe de commandes. Notez que la fonction <time> doit être placée à la fin de la ligne et est associée à l'ensemble du mouvement, tandis que la commande <speed> peut être associée à chaque servo.

Exemple 1 : #5 P1600 #10 P750 T2500 <cr>

L'exemple ci-dessus déplacera le servomoteur 5 vers la position 1600, le servomoteur 10 vers la position 750, et il faudra 2500 millisecondes (2,5 secondes) pour terminer le mouvement, même si un servomoteur doit parcourir une distance plus longue qu'un autre. Les servomoteurs commenceront et arrêteront de bouger en même temps. Il s'agit d'une commande très puissante. Par exemple, en commandant toutes les jambes d'un robot marcheur avec « Group Move », il est facile de synchroniser des allures complexes. Le même mouvement synchronisé peut également bénéficier au contrôle d'un bras robotisé.

Vous pouvez combiner les commandes de vitesse et de temps si vous le souhaitez. La vitesse de chaque servo sera calculée selon les règles suivantes :

1. Tous les canaux démarreront et termineront le déplacement simultanément.
2. Si une vitesse est spécifiée pour un servo, il ne se déplacera pas plus vite que la vitesse spécifiée (mais il pourrait se déplacer plus lentement si la commande de temps l'exige).
3. Si un temps est spécifié pour le déplacement, alors le déplacement prendra au moins le temps spécifié (mais peut prendre plus de temps si la commande de vitesse l'exige).

Exemple 2 : #5 P1600 #17 P750 S500 #2 P2250 T2000 <cr>

L'exemple fournit 1600uS sur ch5, 750uS sur ch17 et 2250uS sur ch2. Le déplacement complet prendra au moins 2 secondes, mais ch17 ne se déplacera pas plus vite que 500uS par seconde. Le temps réel du déplacement dépendra de la largeur d'impulsion initiale pour ch17. Supposons que ch17 démarre à la position 2000. Il doit alors se déplacer de 1250uS. Comme il est limité à 500uS par seconde, il lui faudra au moins 2,5 secondes, donc le déplacement complet prendra 2,5 secondes. D'un autre côté, si ch17 démarre à la position 1000, il n'a besoin de se déplacer que de 250uS, ce qu'il peut faire en 0,5 seconde, donc le déplacement complet prendra 2 secondes.

Important ! La première commande de positionnement doit être une commande normale « # <ch> P <pw> ».
Étant donné que le contrôleur ne sait pas où se trouve le servo lors de la mise sous tension, il ignorera les commandes de vitesse et de temps jusqu'à ce que la première commande normale soit reçue.

Tout mouvement impliquant plusieurs servos et utilisant le modificateur S ou T est considéré comme un mouvement de groupe, et tous les servos commenceront et s'arrêteront de bouger en même temps. Si vous devez déplacer plusieurs servos à des vitesses différentes, vous devez émettre les commandes séparément.
Décalage de position du logiciel.

Décalage de la position du servo

#<ch> PO <valeur de décalage>... #<ch> PO <valeur de décalage>... <cr>

• <ch> : numéro de canal en décimal (0 à 31) • <décalage
de position> : le décalage de position est limité à -100us à 100us (environ 15°) • <cr> : retour chariot

La commande de décalage de position vous permet de modifier la position centrale (associée à 1500us) d'un ou plusieurs servos via le logiciel dans les 15 degrés du centre absolu. La commande de décalage ne doit être envoyée qu'une seule fois au contrôleur de servo SSC-32U au début du programme. Cependant, lorsque le SSC-32U est éteint, il oublie les décalages de position. Notez que le servo lui-même reste inchangé ; seul le signal envoyé à cette ou ces broches particulières du SSC-32 sera modifié.

L'une des utilisations principales de cette fonction est de centrer/aligner parfaitement la position d'un servo avec la mécanique à laquelle il est connecté. Par exemple, lors de l'assemblage d'un servo sur un système de support, vous pouvez remarquer que le servo est désaligné de quelques degrés par rapport au châssis, alors que votre code exige qu'il soit parfaitement aligné. Bien que la correction mécanique de ce problème soit très difficile, la commande de décalage de position permet un réglage précis. Il est toujours important de construire l'assemblage mécanique aussi près que possible de la position centrée souhaitée avant d'appliquer le décalage du servo.

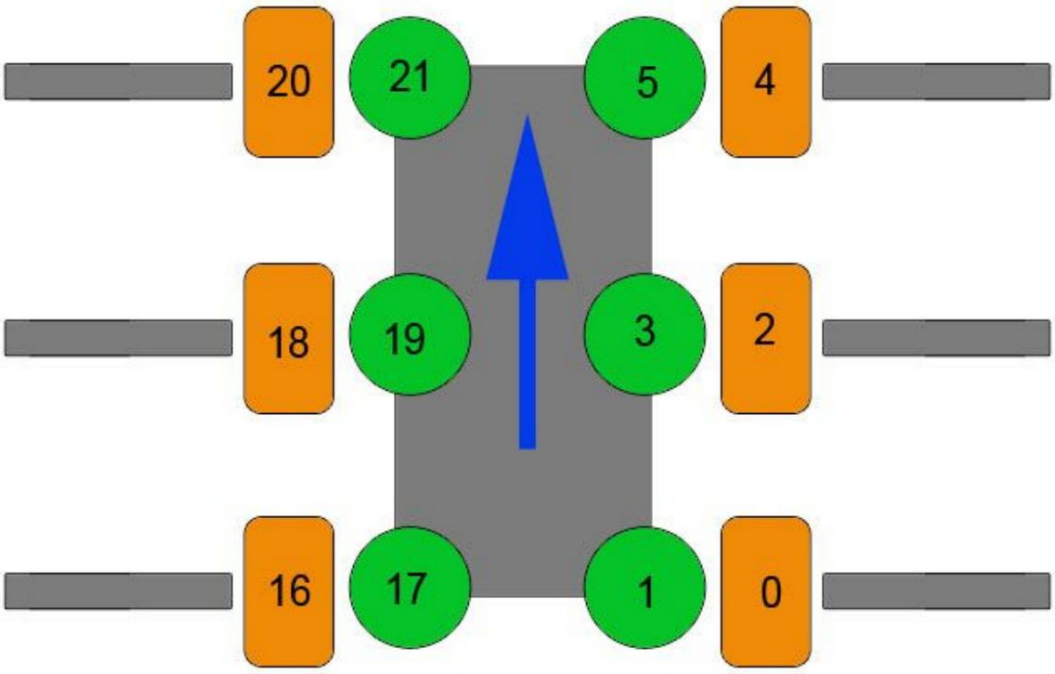
Séquenceur hexapode 12 servos

Le SSC-32U comprend une fonction intégrée pour contrôler facilement 12 degrés de liberté (six jambes, deux degrés de liberté par jambe) robot marcheur hexapode. La fonction nécessite toujours des commandes série à envoyer depuis une source externe (microcontrôleur, ordinateur ou module sans fil).

Les canaux servo suivants sont utilisés pour le séquenceur hexadécimal (vertical / horizontal se réfèrent au mouvement):

0	Arrière droit vertical	16	Arrière gauche vertical
1	Arrière droit horizontal	17	Arrière gauche horizontal
2	Centre droit vertical	18	Gauche Centre Vertical
3	Centre droit horizontal	19	Gauche Centre Horizontal
4	Avant droit vertical	20	Avant gauche vertical
5	Avant droit horizontal	21	Avant gauche horizontal

Affectation des broches du servomoteur Hexapode

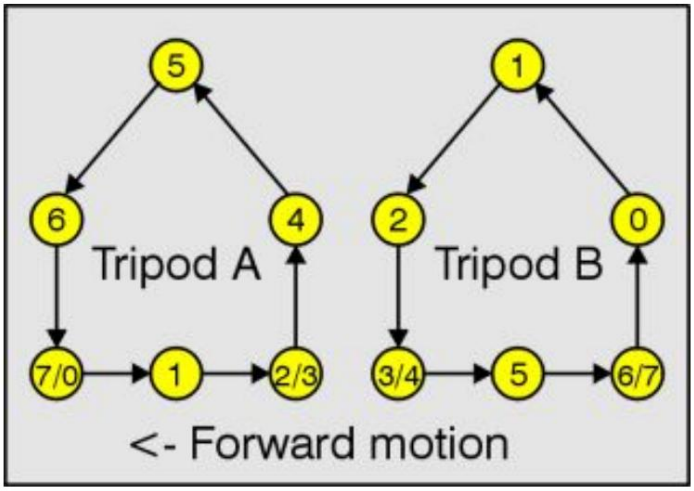


Affectation visuelle des broches pour le séquenceur Hexapod

La séquence de marche se compose de 8 états, numérotés de 0 à 7. La démarche utilise un trépied alterné
démarche (trois jambes en haut, trois jambes en bas) jusqu'à la marche, avec six emplacements possibles pour chaque pied.
Les trépieds sont étiquetés Trépied A et Trépied B.

- Le trépied A se compose de {jambe avant gauche, jambe arrière gauche, jambe centrale droite}
- Le trépied B se compose de {jambe centrale gauche, jambe avant droite, jambe arrière droite}

Ils sont définis ci-dessous :

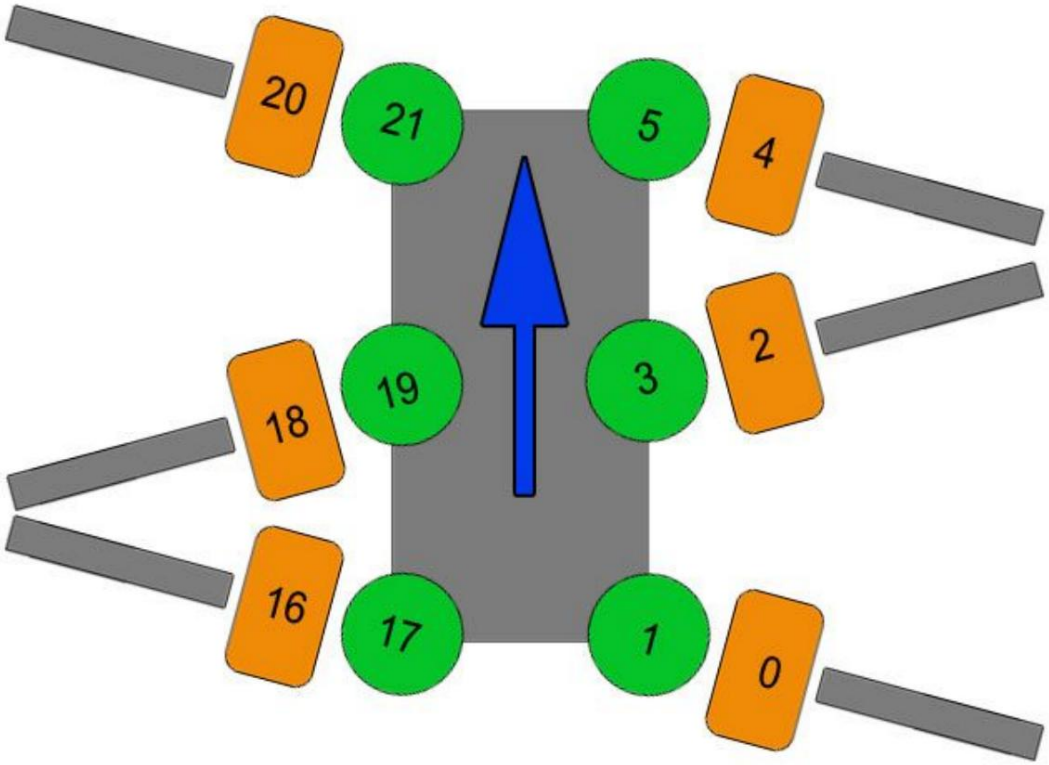


Vue de la jambe de côté pour chaque groupe de trois jambes

Lors de la marche, les jambes passent par 6 points : (Bas devant), (Bas centre), (Bas arrière), (Milieu
Arrière), (Centre haut) et (Milieu avant). « Centre » fait référence au point médian entre l'avant et
Positions arrière.

État	Servomoteur vertical (Trépied A)	Servomoteur horizontal (Trépied A)	Servomoteur vertical (Trépied B)	Servomoteur horizontal (Trépied B)
0	Faible	De l'avant vers le centre	Moyen à élevé	De l'arrière vers le centre
1	Faible	Du centre vers l'arrière	Élevé à moyen	Du centre vers l'avant
2	Faible	Arrière	Moyen à bas	Devant
3	Faible à moyen	Arrière	Faible	Devant
4	Moyen à élevé	De l'arrière vers le centre	Faible	De l'avant vers le centre
5	Élevé à moyen	Du centre vers l'avant	Faible	Du centre vers l'arrière
6	Moyen à bas	Devant	Faible	Arrière
7	Faible	Devant	Faible à moyen	Arrière

Position et mouvement du pied pendant le mouvement



Configuration de la démarche de marche

LH <arg>, LM <arg>, LL <arg>

Définissez la valeur des servos verticaux sur le côté gauche de l'hexapode. LH définit la valeur haute, c'est-à-dire la largeur d'impulsion pour élever la jambe à sa hauteur maximale ; LM définit la valeur moyenne ; et LL définit la valeur basse. La plage valide pour les arguments est de 500 à 2500uS.

RH <arg>, RM <arg>, RL <arg>

Définissez la valeur des servos verticaux sur le côté droit de l'hexapode. RH définit la valeur haute, c'est-à-dire la largeur d'impulsion pour élever la jambe à sa hauteur maximale ; RM définit la valeur moyenne ; RL définit la valeur basse. La plage valide pour les arguments est de 500 à 2500uS.

VS <arg>

Définit la vitesse de déplacement des servos verticaux. Tous les mouvements de servos verticaux utilisent cette vitesse. La plage valide est de 0 à 65 535 µs/s.

LF <arg>, LR <arg>

Définissez la valeur des servos horizontaux sur le côté gauche du robot. LF définit la valeur avant, c'est-à-dire la largeur d'impulsion pour déplacer la jambe vers la position avant maximale ; LR définit la valeur arrière. La plage valide pour les arguments est de 500 à 2500uS.

RF <arg>, RR <arg>

Définissez les valeurs des servos horizontaux sur le côté droit du robot. RF définit la valeur avant, c'est-à-dire la largeur d'impulsion pour déplacer la jambe vers la position avant maximale ; RR définit la valeur arrière. La plage valide pour les arguments est de 500 à 2500uS.

HT <arg>

Définit le temps de déplacement entre les positions horizontales avant et arrière. La plage valide pour l'argument est comprise entre 1 et 65 535 uS.

XL <arg>, XR <arg>

Définissez le pourcentage de déplacement pour les jambes gauche et droite. La plage valide est de -100 % à 100 %. Les valeurs négatives entraînent le déplacement des jambes latérales en sens inverse. Avec une valeur de 100 %, les jambes se déplaceront entre les positions avant et arrière. Des valeurs inférieures entraînent un déplacement proportionnellement plus petit, mais toujours centré. La vitesse des déplacements horizontaux est ajustée en fonction des commandes XL et XR, de sorte que le temps de déplacement reste le même.

XS <arg>

Définissez le pourcentage de vitesse horizontale pour toutes les jambes. La plage valide est de 0 % à 200 %. Avec une valeur de 100 %, le temps de déplacement horizontal sera la valeur programmée à l'aide de la commande HT. Des valeurs plus élevées réduisent proportionnellement le temps de déplacement ; des valeurs plus faibles l'augmentent. Une valeur de 0 % arrêtera le robot sur place. Le séquenceur hexadécimal ne démarrera pas tant que la commande XS n'aura pas été reçue.

XSTOP

Arrêtez le séquenceur hexadécimal. Remettez tous les servos en fonctionnement normal.

1. Lorsqu'un servo horizontal se déplace, sa vitesse sera ajustée en fonction des largeurs d'impulsion avant/arrière, du pourcentage XL/XR et du pourcentage XS. Quelle que soit la distance de déplacement de l'avant vers l'arrière (ajustée par XL/XR), le temps de déplacement total sera le HT divisé par le pourcentage XS.

2. Lorsqu'un servo vertical se déplace de Low à Mid ou de Mid à Low, il se déplace à la vitesse spécifiée par la commande VS. Lorsqu'un servo vertical se déplace de Mid à High ou de High à Mid, la vitesse verticale est ajustée de sorte que les mouvements horizontaux et verticaux se terminent en même temps.

3. N'importe laquelle des commandes du séquenceur hexadécimal peut être émise pendant que le séquenceur fonctionne. Elles entreront en vigueur immédiatement.

Exemples de séquenceur hexadécimal :
« LH1000 LM1400 LL1800 RH2000 RM1600 RL1200 VS3000 <cr> »

Définit les paramètres du servo vertical.
"LF1700 LR1300 RF1300 RR1700 HT1500 <cr>"

Définit les paramètres du servo horizontal.
« XL50 XR100 XS100 <cr> »

Provoque le virage progressif à gauche à 100 % de la vitesse (et démarre le séquenceur s'il n'est pas déjà démarré).

« XL-100 XR 100 XS 50 <cr> »

Provoque une rotation à gauche sur place à 50 % de la vitesse.
"XSTOP <cr>"

Arrête le séquenceur et permet de contrôler les canaux servo 0-5, 16-21 à l'aide des commandes servo normales.

Interroger l'état du séquenceur hexadécimal.
XQ <cr>

Renvoie 1 chiffre représentant l'état du séquenceur hexadécimal et le pourcentage approximatif de mouvement dans l'état. Le quartet haut sera compris entre « 0 » et « 7 » et le quartet bas sera compris entre « 0 » et « 9 ». Par exemple, si le séquenceur est à 80 % de l'état 5, il renverra la valeur 58 (hexadécimale).

Fonctions avancées

Annuler la sortie

<ch> P <pw> ….<esc>

- <esc>: Annuler l'action en cours, ASCII 27

Si vous souhaitez annuler une commande, ajoutez un <esc> à la fin de la ligne :

Sortie discrète Les
broches IO du SSC-32 peuvent être utilisées pour fournir des signaux HAUT (5 V) ou BAS (0 V). Notez que vous ne devez PAS utiliser cette fonction avec des servos RC standard.

<ch> <niv> ... # <ch> <niv> <cr>

- <ch> : Numéro de canal en décimal, 0-31 • <|v|> :
  Niveau logique du canal, soit « H » pour haut, soit « L » pour bas • <cr> : Caractère de
  retour chariot, ASCII 13

Les sorties du SSC-32 proviennent de quatre puces de registre à décalage de 8 bits. Il existe quatre banques de sorties 8 bits comme indiqué 0-7, 8-15, 16-23 et 24-32. Les sorties peuvent absorber ou générer jusqu'à 20 mA par broche de sortie, mais un maximum de 70 mA par banque doit être respecté. Le canal passera au niveau indiqué dans les 20 ms suivant la réception du retour chariot.

Exemple : #3H #4L <cr>

Cet exemple produira une sortie haute (+5 V) sur le canal 3 et une sortie basse (0 V) sur le canal 4.

Sortie d'octets

<banque> : <valeur> <cr>

- <bank>: (0 = Broches 0-7, 1 = Broches 8-15, 2 = Broches 16-23, 3 = Broches 24-31) •
  <value>: Valeur décimale à envoyer à la banque sélectionnée (0-255), Bit 0 = LSB de la banque • <cr>:
  Caractère de retour chariot, ASCII 13

Cette commande permet d'écrire 8 bits de données binaires en une seule fois. Toutes les broches de la banque sont mises à jour simultanément. Les banques seront mises à jour dans les 20 ms suivant la réception du retour chariot. Notez que les deux points (:) sont obligatoires.

Exemple : #3:123 <cr>

Cet exemple génère la valeur 123 (décimal) vers la banque 3. 123 (déc) = 01111011 (bin), et la banque 3 est associée aux broches 24 à 31. Cette commande génère donc un « 0 » vers les broches 26 et 31, et génère un « 1 » vers toutes les autres broches.

Statut du mouvement de la requête

Q <cr>

Cela renverra un « . » si le mouvement précédent est terminé, ou un « + » s'il est toujours en cours.
Il y aura un délai de 50 µS à 5 mS avant l'envoi de la réponse.

Largeur d'impulsion de requête

QP <arg> <cr>

Cela renverra un seul octet (au format binaire) indiquant la largeur d'impulsion du servo sélectionné avec une résolution de 10 µS. Par exemple, si la largeur d'impulsion est de 1 500 µS, l'octet renvoyé sera 150 (binaire).

Plusieurs servos peuvent être interrogés dans la même commande. La valeur de retour sera d'un octet par servo. Il y aura un délai d'au moins 50 µS à 5 mS avant que la réponse ne soit envoyée. En général, la réponse sera lancée dans les 100 µS.

Entrée numérique

ABCDEF AL BL CL DL EL FL<cr>

A, B, C, D, E, F sont des lectures d'entrée normales. Elles lisent la valeur de l'entrée sous forme de valeur binaire. Elles renvoient un "0" ASCII si l'entrée est basse (0 V) ou un "1" ASCII si l'entrée est haute (+ 5 V). Comme vous pouvez le voir, les broches G et H ne sont pas incluses (elles sont uniquement analogiques).

AL, BL, CL, DL, EL, FL sont des lectures d'entrée à verrouillage. Elles renvoient la valeur de l'entrée sous forme de « 0 » ASCII si l'entrée est basse (0 V) ou si elle est basse depuis la dernière commande *L. Elles renvoient une valeur haute (+ 5 V) si l'entrée est haute et n'est jamais passée à l'état bas depuis la dernière commande *L. En termes simples, elles renvoient une valeur basse si l'entrée passe à l'état bas. La lecture de l'état réinitialise simplement le verrou.

Ces entrées ont une faible résistance de rappel (~50k) qui est activée lorsqu'elles sont utilisées comme entrées. Elles sont vérifiées environ toutes les 1 ms et sont débarrassées des rebonds pendant environ 15 ms. La valeur logique des commandes de lecture ne sera pas modifiée tant que l'entrée n'aura pas été au nouveau niveau logique en continu pendant 15 ms. Les commandes d'entrée numérique de lecture peuvent être regroupées en une seule lecture, jusqu'à 8 valeurs par lecture. Elles renverront une chaîne avec un caractère par entrée sans espace.

Exemple : ABE DL <cr>

Entrées analogiques

VA VB VC VD VE VF VG VH <cr>

Les broches étiquetées A à H sont des broches d'entrée/sortie analogiques qui peuvent être utilisées pour lire des capteurs, pour piloter des LED basse consommation, etc.

Exemple : « VA VB VC VD <cr> »

VA, VB, VC et VD lisent la valeur de l'entrée sous forme analogique. Ils renvoient un seul octet avec la valeur 8 bits (binaire) de la tension sur la broche.

Lorsque les entrées ABCD sont utilisées comme entrées analogiques, la résistance de rappel interne est désactivée. Les entrées sont filtrées numériquement pour réduire l'effet du bruit. Les valeurs filtrées se stabiliseront à leurs valeurs finales dans les 8 ms suivant un changement. Une valeur de retour de 0 représente 0 VCC. Une valeur de retour de 255 représente +4,98 VCC. Pour convertir la valeur de retour en tension, multipliez par 5/256. À la mise sous tension, les entrées ABCD sont configurées pour une entrée numérique avec résistance de rappel. La première fois qu'une commande V* est utilisée, la broche sera convertie en analogique sans résistance de rappel. Le résultat de cette première lecture ne renverra pas de données valides.
Lire l'exemple d'entrée analogique : « VA VB <cr> »

Cet exemple renverra 2 octets avec les valeurs analogiques de A et B. Par exemple, si la tension sur la broche A est de 2 Vcc et la broche B de 3,5 Vcc, la valeur de retour sera les octets 102 (binaire) et 179 (binaire).

Baud

Le SSC-32U est livré avec un débit en bauds par défaut de 9600. Il prend également en charge le réglage du débit en bauds à l'aide du bouton-poussoir intégré. Pour régler le débit en bauds : 1.

Appuyez sur le bouton et maintenez-le enfoncé. Au début, les LED s'allument pour indiquer le débit en bauds actuel. a.
9600 (vert) b. 38400
(rouge) c. 115200
(vert et rouge) d. Débit en bauds non
standard (pas de LED)

2. Après 2 secondes, les LED commenceront à alterner, indiquant que vous pouvez modifier le débit en bauds.
3. Relâchez le bouton.
4. Appuyez sur le bouton pour parcourir les débits en bauds décrits à l'étape 1.
5. Une fois que vous avez sélectionné le débit en bauds souhaité, ne faites rien ; après 5 secondes, les LED reviendront au mode normal et le nouveau débit en bauds sera écrit dans l'EEPROM.

Le registre R4 contient désormais le débit en bauds. En plus de cette manière physique de définir le débit en bauds, il peut être écrit via l'ordinateur à l'aide de la commande ci-dessous.
Le débit en bauds peut être défini sur des valeurs non standard si vous le souhaitez en écrivant dans R4. Afin d'adapter les débits en bauds à une valeur de 16 bits, il stocke la valeur divisée par 10 ; ainsi, un débit de 9 600 bauds serait

stocké sous la forme 960. Par exemple, pour régler sur 2400 bauds, exécutez la commande R4=240. Dans ce cas, appuyer sur le bouton n'allumera pas les LED, ce qui indique que le débit en bauds n'est pas standard. (Les LED clignoteront brièvement lorsqu'elles seront enfoncées, juste pour que vous sachiez que quelque chose s'est passé.)

Vous pouvez relire le débit en bauds actuel en entrant simplement la commande « R4 » suivie d'un retour chariot.

« R4<cr> » donnera « 240 » si le débit en bauds est de 2 400.

Micrologiciel

Pour mettre à jour le firmware, utilisez l'utilitaire gratuit SSC-32 Servo Sequencer Utility (RB_Dsp-07)

Registres SSC-32

Inscription Informations générales

Numéro Nom		Minimum Maximum Défaut			Description
0	Activer	0	65535	0	Un champ de bits (16 bits) qui permet diverses fonctionnalités du SSC-32.
1	Délai de transmission	0	65535	600	Le délai, en microsecondes, avant la transmission le premier octet d'une réponse du SSC-32.
2	Transmission du rythme	0	65535	70	Le délai, en microsecondes, entre les octets d'un réponse du SSC-32.
3-31	(Réservé)	--	--	--	--
32-63	Impulsion initiale Compenser	-100	100	0	La valeur initiale du décalage d'impulsion (PO) pour chaque servo. Le registre 32 correspond au servo #0, registre 33 au servo #1, etc.
64-95	Impulsion initiale Largeur	0	65535	1500	La valeur initiale de la largeur d'impulsion pour chaque servo. Le registre 64 correspond au servo #0, registre 65 au servo #1, etc. Une valeur de 0 quitte la sortie du servo à une logique continue « 0 » ; une valeur de 65535 quitte la sortie du servo à un régime continu logique « 1 ». Toutes les autres valeurs sont limitées à la plage 500 - 2500 microsecondes.
96-255	(Réservé)	--	--	--	--

Remarque : les registres 0 à 15 sont destinés à une utilisation globale, affectant toutes les opérations du SSC-32 ; les registres Les 32-255 sont destinés à la configuration individuelle des canaux servo et sont donc répartis en groupes de 32 registres.

Définitions des bits du registre d'activation (R0)

Peu	Nom	Définition
15 (msb)	Mondial Désactiver	Si « 1 », désactive toutes les fonctionnalités contrôlées par le registre d'activation. Si « 0 », les valeurs de bits individuelles seront utilisées pour activer ou désactiver les fonctionnalités.
14-4	(Réservé)	--
3	Impulsion initiale Activation de la largeur	Si « 1 », active les valeurs du registre de largeur d'impulsion initiale au démarrage. Si « 0 », la valeur par défaut de 0 sera utilisée.
2	Impulsion initiale Activation du décalage	Si « 1 », active les valeurs du registre de décalage d'impulsion initial au démarrage. Si « 0 », la valeur par défaut de 0 sera utilisée.
1	TX Retard/rythme Activer	Si « 1 », active les valeurs de délai de transmission et de rythme de transmission. Si « 0 », les valeurs par défaut de 600 uS et 70 uS seront utilisées.
0 (lsb)	Chaîne de démarrage Activer	Si « 1 », active l'exécution de la chaîne de démarrage lorsque l'alimentation est appliquée au SSC-32. Si « 0 », la chaîne de démarrage ne sera pas exécutée.

S'inscrire en lecture/écriture

Commande	Argument	Description	Exemples
S'inscrire écrivez : R <r> = <n> <cr>	r = numéro d'enregistrement, 0-255 n = valeur	Programme la valeur d'un registre. Les espaces sont facultatifs autour du numéro de registre et de la valeur.	R0=1023 <cr> R32 = -50 <cr>
S'inscrire lire: R <r> <cr>	r = numéro d'enregistrement, 0-255	Affiche la valeur d'un registre, suivie d'un retour chariot. La valeur affichée est au format ASCII et se termine par un retour chariot.	R0 <cr> résultat : 1023<cr> R32 <cr> résultat : -50<cr>
Définir les valeurs par défaut : RDFLT <cr>	aucun	Définit tous les registres à leurs valeurs par défaut. Lorsque la commande une fois terminé, le SSC-32 transmettra la chaîne OK<cr>.	RDFLT <cr> résultat : OK<cr>

L'exécution de la commande RDFLT peut prendre plus d'une seconde. Elle ne doit pas être invoquée lorsqu'un joueur effectuant un mouvement ou une séquence chronométrée est actif. Aucune écriture de registre ne doit être effectuée tant que la commande RDFLT n'est pas terminée (comme indiqué par la réponse « ok »).

Si plusieurs commandes R= sont envoyées par le logiciel, il est recommandé que le logiciel lise la valeur de chaque registre après son écriture. Cela permettra de garantir que chaque écriture de registre est terminée avant le début de la suivante.

Si une commande RDFLT ou R= est en cours d'exécution, ne mettez pas le SSC-32 hors tension tant que la commande n'est pas terminée. Pour déterminer si la commande est terminée, lisez une valeur de registre. Chaque fois qu'un registre est écrit, les emplacements EEPROM utilisés pour stocker la valeur subissent une légère usure. Le nombre maximum d'écritures typique est de 100 000. N'écrivez pas votre logiciel pour modifier rapidement les valeurs des registres, car vous risqueriez de provoquer une usure permanente de l'EEPROM du processeur ATMega168.

Commandes de registre diverses

Description de l'argument de commande			Exemples
ARRÊTEZ <n> <cr>	0-31	Arrête immédiatement le servo spécifié à sa position actuelle. Un espace est facultatif avant le numéro du servo.	STOP0 <cr> ARRÊT 31 <cr>

Si le servo fait partie d'un mouvement chronométré, les autres servos continueront de se déplacer et une commande de requête indiquera que le mouvement continue jusqu'à ce que le temps total du mouvement d'origine soit écoulé. Cela est vrai même si tous les servos du mouvement d'origine sont arrêtés.

Les commandes EER et EEW ne fonctionnent plus pour accéder à l'EEPROM interne. Elles sont remplacées par les commandes Register Read/Write et Startup String. EER et EEW continuent de fonctionner pour l'EEPROM externe.

Chaînes de démarrage

Description/exemples d'arguments de commande		
Supprimer personnages: SSDEL <n> <cr>	0-255	Supprime <n> caractères de la fin de la chaîne de démarrage. Si <n> est supérieur à la longueur de la chaîne de démarrage, SSDEL supprime la chaîne entière. SSDEL 5 <cr> - Suppression des 5 derniers caractères de la chaîne de démarrage SSDEL 255 <cr> - Supprime toute la chaîne de démarrage
Enchaîner: SSCAT <chaîne> <cr>	Jusqu'à 100 ASCII personnages	Concatène <string> à la chaîne de démarrage actuelle. L'espace vide qui suit immédiatement « SSCAT » est ignoré, mais tous les autres espaces font partie de la chaîne. La chaîne se termine par un retour chariot et ne peut pas contenir de retour chariot intégré. Les commandes de la chaîne de démarrage se terminent par un point-virgule (y compris la dernière commande). SSCAT #0P1000#1P2000T3000;<cr> SSCAT PL0 SQ5 SM50;<cr>
Afficher la chaîne de démarrage : SS <cr>	aucun	Affiche la chaîne de démarrage entière, entourée de guillemets et suivie d'un retour chariot. SS <cr> résultat : « #0P1000#1P2000T3000 ;PL0 SQ5 SM50 ; »<cr>

La chaîne de démarrage programmée est exécutée à la mise sous tension du SSC-32, si le bit d'activation de la chaîne de démarrage est défini dans le registre d'activation. La chaîne de démarrage est exécutée après l'application de toutes les valeurs de registre (par exemple, la largeur d'impulsion initiale). La longueur totale maximale de la chaîne de démarrage est de 100 caractères ASCII. Tous les caractères supplémentaires seront ignorés.

Les commandes suivantes ne doivent pas être utilisées dans une chaîne de démarrage : EER, EEW, R=, SSCAT, SSDEL. L'exécution de la commande SS peut prendre des centaines de millisecondes, selon le débit en bauds. Elle ne doit pas être invoquée lorsqu'un joueur de mouvement ou de séquence chronométré est actif.
L'exécution de la commande SSCAT peut prendre des centaines de millisecondes. Elle ne doit pas être invoquée lorsqu'un joueur effectuant un mouvement ou une séquence chronométrée est actif.

Si une commande SSDEL ou SSCAT est en cours d'exécution, ne mettez pas le SSC-32 hors tension tant que la commande n'est pas terminée. Pour déterminer si la commande est terminée, envoyez une commande SS et attendez la réponse. Chaque fois que la chaîne de démarrage est modifiée, les emplacements EEPROM utilisés pour stocker la valeur subissent une légère usure. Le nombre maximal d'écritures typique est de 100 000. N'écrivez pas votre logiciel pour modifier rapidement la chaîne de démarrage, car vous risquez de provoquer une usure permanente de l'EEPROM dans le processeur ATmega168.

Exemples de chaînes de démarrage	
Commande	Résultat
SSDEL 255 <cr> SS <cr>	""<cr>
SSCAT #0P2000T5000;<cr> SS <cr>	"#0P2000T5000;"<cr>
SSCAT XXXX<cr> SSC <cr>	« #0P2000T5000 ; XXXX »<cr>
SSDEL 4 <cr> SS <cr>	"#0P2000T5000;"<cr>
SSDEL 6 <cr> SS <cr>	"#0P2000"<cr>
SSCAT #1P1000T4000;PL0SQ5;<cr> SS <cr>	"#0P2000#1P1000T4000;PL0S Q5;"<cr>

Exemples supplémentaires	
Commande	Résultat
RDFLT	Régler tous les registres sur les valeurs par défaut
SSDEL 255	Effacer la chaîne de démarrage
R0	Registre d'affichage 0
R0=2 R1=2000 R2=1000	Réglez le délai d'émission sur 2 000 uS et la stimulation d'émission sur 1 000 uS. (R0 = 2 : le bit 1 de R0 active le délai/la stimulation d'émission.)
R0=12 R32=50 R64=1000	Réglez le décalage d'impulsion pour le servo de 0 à 50 et la largeur d'impulsion initiale à 1000. (Les bits 2 et 3 de R0 activent le décalage d'impulsion et la largeur d'impulsion.)
R0=13 SSDEL 255 Numéro de série SSCAT 0P1500T5000;	Déplacez lentement R0 jusqu'à une largeur d'impulsion de 1 500 au démarrage. Supposons que la largeur d'impulsion initiale soit définie comme dans l'exemple précédent. (Le bit 0 de R0 active la chaîne de démarrage.)
SS	Affiche la chaîne de démarrage actuelle.