Encore une présentation des servomoteurs Feetech

Débuté par Keuronde, aoÃ»t 05 2025 06:02

Feetech, *Servomoteur,*

Keuronde

Posté 05 aoÃ»t 2025 - 06:02

Alors, pourquoi "encore" une présentation ? Parce que EVRIS en a déjà faite une première ici (https://www.robot-maker.com/forum/topic/14804-presentation-et-fonctionnement-des-servomoteurs-feetech/) . Qu'il y a un article de Marilux sur comment s'interfacer avec les Feetech (https://www.robot-maker.com/forum/tutorials/article/168-brancher-et-controler-le-servomoteur-feetech-sts3032-360/) .

Et que malgré ça, j'ai transpiré un peu pour me servir des Feetech. J'ai donc deux possibilités, vous présenter rapidement ce qui m'a posé soucis et ma solution ou repartir de zéro et faire une présentation en profondeur des Feetech. J'opte pour la seconde solution. Attendez-vous alors a certaines re-dites par rapport aux articles cités plus haut.

Introduction

Les servomoteurs de modélisme

Les servomoteurs de modélisme sont des actionneurs assez intéressants car ils sont simples et sont commandés en position. Contrairement à un moteur à courant continu, où le moteur tourne dès que vous appliquez une tension, un servomoteur de modélisme va "attendre", en plus de l'alimentation, une consigne de position. C'est le servomoteur, qui en fonction de sa position et de la consigne envoyée, va commander son moteur pour atteindre la position demandée.

Pour les mettre en œuvre, le schéma est assez simple. Sur les trois broches, deux servent à l'alimentation et la troisième est directement reliée au signal du microcontrôleur. Contrairement à un moteur à courant continu, vous n'avez pas besoin de pont en H pour les piloter. Tout est intégré dans le servomoteur.

Les servomoteurs de modélisme présentent quelques limitations :
- Leur course est généralement limitée à 180°
- Il faut une broche de microcontrôleur par servomoteur. C'est peut-être un détail, mais ça peut empêcher de réutiliser une carte électronique d'une année sur l'autre.
- Difficile de piloter finement la vitesse du servomoteur
- Et surtout, le servomoteur ne renvoie pas d'information sur son état

Les servomoteurs Feetech

- Leur course est réglable, et illimitée dans certains mode d'utilisation
- Ils sont chaînables, sur un port vous pouvez brancher tous vos servomoteurs
- Leur vitesse et leur accélération sont paramétrables
- Ils renvoient des informations détaillées sur leur état, position, vitesse, effort...

Ces servomoteurs existent en 2 gammes. L'une communique avec un protocole RS485, ce qui est certainement un inconvénient pour un petit projet. En effet, vous aurez besoin d'une carte d'adaptation car vous ne trouver pas (ou très difficilement) de microcontrôleurs qui intègrent le protocole RS485. L'avantage est que ce protocole, utilisant un signal différentiel, est résistant aux perturbations.

L'autre gamme utilise un signal TLL, en half duplex. Cela signifie que l'émission et la réception des données se fait sur la même broche. Si pour ce protocole, une grosse bidouille logicielle semble possible, le plus simple sera d'utiliser une carte de conversion.

Le fabriquant vend une carte de conversion UART <-> RS485 ou UART <-> TTL Half Duplex pour simplifier l'intégration de servomoteur de vos projets que vous trouverez chez Robot Maker. Nous utiliserons celle-ci : https://www.robot-ma...er-mini-69.html (https://www.robot-maker.com/shop/drivers-d-actionneurs/69-ttlinker-mini-69.html)

mais vous pouvez utiliser celle-là si vous souhaitez utiliser un ordinateur pour communiquer avec vos servomoteur : https://www.robot-ma...-urt-1-460.html (https://www.robot-maker.com/shop/alimentation/460-convertisseur-feetech-ttlrs485-fe-urt-1-460.html)

Première étape - S'installer

Câblage

Vous devez connecter sur la carte d'adaptation :

- L'alimentation de puissance (5V)
- L'alimentation logique (3,3V)
- Les signaux UART (RX/TX) sans les croiser

Installation de l'IDE

On va supposer que vous avez un IDE d'installé, que ce soit celui d'Arduino, VSCode + PlatformIO, ou autre.

Récupération de la bibliothèque

Robot Maker fourni une bibliothèque servomoteurs feetech (https://github.com/Robot-Maker-SAS/FeetechServo) qui offre les principales fonctions. Nous en aurons besoin dès le début car la bibliothèque permet de construire les trames.

Adaptation de la liaison série

Même avec Arduino, la configuration de la liaison série peut demander un peu de travail. En général, Serial.begin() démarre le port série qui communique avec

- Pour les cartes officielles, la documentation est ici : https://docs.arduino...ication/serial/ (https://docs.arduino.cc/language-reference/en/functions/communication/serial/)
- Sur un ESP32, vous devez également préciser les broches à utiliser : Serial1.begin(1000000, SERIAL_8N1, RX, TX); RX et TX étant les numéros des broches. Ce sont parfois des constantes définies par le fichier décrivant votre carte.
- Sur un RP2040 (Raspberry Pi Pico), PlatformIO ne laisse pas le choix, TX sera la broche 0, RX la broche 1.

Dans tous les cas, vous devez donnez le baudrate, qui est de 1000000 pour communiquer avec les Feetechs

Ping du servomoteur

Voici un programme qui teste la présence d'un servomoteur qui aurait l'adresse 1 (notez que c'est l'adresse par défaut).
Si ça marche, vous être prêt à passer à la suite !

```
#include <Arduino.h>

#include "SCServo.h"
SMS_STS sms_sts;


void setup() {
 // Configure la led en sortie
 pinMode(LED_BUILTIN, OUTPUT);


 // On ouvre 2 liaisons séries, 1 pour communiquer avec le PC, 1 pour les Feetec
 Serial.begin(115200);
 Serial1.begin(1000000);
 sms_sts.pSerial = &Serial1;
}

// the loop routine runs over and over again forever:
void loop() {
 digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage
 test_ping();
 digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage
 test_ping();
}

void test_ping()
{
 int ID = sms_sts.Ping(1);
 if(ID!=-1){
  Serial.print("Servo ID:");
  Serial.println(ID);
  delay(100);
 }else{
  Serial.print("Ping servo ID error!\n");
  delay(2000);
 }
}
```

Alors, ce n'est que l'intro, dans les épisodes à venir, nous pourrions avoir :

- Le détail du protocole

N'hésitez pas à faire part de vos remarques, critiques, propositions d'amélioration ici !

Keuronde

Posté 08 aoÃ»t 2025 - 02:57

Si vous êtes arrivé jusqu'ici, vous avez :
- installé l'IDE
- installé la bibliothèque
- réalisé le câblage

Nous allons pouvoir nous pencher sur le fonctionnement détaillé de ces servomoteurs Feetech.

Ce qui suit est valable pour un type de servomoteur - ceux à potentiomètre.

Le protocole

Penchons nous un peu sur le protocole de communication.
Le protocole de communication est défini par Feetech, la documentation n'est pas simple, ni à trouver, ni à lire...
Pour la trouver, le lien direct est ici : https://www.feetechr... (https://www.feetechrc.com/Data/feetechrc/upload/file/20240702/%E8%88%B5%E6%9C%BA%E5%8D%8F%E8%AE%AE%E6%89%8B%E5%86%8C-%E7%94%B5%E4%BD%8D%E5%99%A8%E7%89%88%E6%9C%AC.pdf)

Quand à la lire, voici les grandes lignes. Il y a deux types de trame, les instructions et les réponses. Une instructions est structurée ainsi :

- Entête : 2 octets (0xFF 0xFF)
- L'identifiant du servomoteur : 1 octet (1 à 255)
- La taille du message : 1 octect
- L'instruction (voir plus bas) : 1 octet
- Les paramètres de l'instruction : 1 ou plusieurs octets
- La somme de contrôle : 1 octet

La réponse du servomoteur est structurée ainsi :

- Entête : 2 octets (0xFF 0xFF)
- L'identifiant du servomoteur : 1 octet (1 à 255)
- La taille du message : 1 octet
- Le status du servomoteur : 1 octet (0 si OK)
- Les valeurs de la réponse : 1 ou plusieurs octets
- La somme de contrôle : 1 octet

Les instructions sont peu nombreuses :

- 0x01 : Ping
- 0x02 : Lire registre
- 0x03 : Écrire registre
- 0x04 : Écriture tamporisé
- 0x05 : Execute les instructions tamporisés
- 0x06 : Écriture synchronisée (pour envoyer la même instruction à plusieurs servomoteurs - aux identifiants consécutifs)
- 0x07 : Réinitialisation des paramètres d'usine

Pour comprendre la logique de fonctionnement des servomoteur Feetech, nous n'avons besoin de connaître que trois instructions : ping (que nous avons déjà utilisée sans le savoir), lire registre et écrire registre.

La tables des registres

C'est donc dans la fonction des registres que réside tous les secrets du servomoteur Feetech.

Feetech met encore une fois à disposition sur son site la documentation (https://www.feetechr...-agreement.html (https://www.feetechrc.com/letter-of-agreement.html)).
Mais encore une fois, c'est pas simple à trouver, et en chinois. Le document qui nous intéresse s'appelle "Servo Protocole Memory Table" (https://www.feetechr... (https://www.feetechrc.com/Data/feetechrc/upload/file/20240702/%E8%88%B5%E6%9C%BA%E5%8D%8F%E8%AE%AE%E5%86%85%E5%AD%98%E8%A1%A8-%E7%94%B5%E4%BD%8D%E5%99%A8%E7%89%88%E6%9C%AC.xlsx))

La mémoire est découpée en 4 parties :

- EPROM en lecture seule. Elle contient quelques informations liés à la fabrication du servomoteur (Version logicielle et matérielle)
- EPROM en lecture-écriture. Elle contient des paramètres que ne devraient être modifiés que rarement (tel que l'identifiant du servomoteur ou la vitesse du protocole de communication). Cette mémoire est protégée, vous devrez modifier une valeur dans la partie suivante pour pouvoir la modifier.
- SRAM en lecture-écriture. Elle contient les valeurs qui seront modifiée couramment pendant l'utilisation normale du servomoteur (Position cible, vitesse)
- SRAM en lecture seule. Elle contient l'état actuel du servomoteur (position, vitesse, tension, température, erreur, courant)

EPROM en lecture seule

Adresse - Fonction
0x00 - Version logicielle (majeur)
0x01 - Version logicielle (mineur)
0x02 - Réserve
0x03 - Version matérielle (majeur)
0x04 - Version matérielle (mineur)

EPROM en lecture et écriture

Adresse - Fonction
0x05 - Identifiant, doit être unique sur le bus. L'identifiant 254 est réservé.
0x06 - Débit du protocole de communication, valeur de 0 à 7, respectivement 1000000, 500000, 250000, 128000, 115200, 76800, 57600 et 38400 bauds. Par défaut, 0 : 1 Mbauds.
0x07 - Délais de retour ou temps entre la réception de la commande et le début de la réponse, par pas de 2 µs. Par défaut : 250 (soit 500 µs).
0x08 - Réponse systématique. 0, ne répond qu'au demande de Ping et aux

0x0B - Butée maximale de l'angle. (2 octets - de 0 à 4095)
0x0D - Limite de température maximale, 70° par défaut.
0x0E - Tension d'entrée maximale (en 0,1 V)
0x0F - Tension d'entrée minimale (en 0,1 V)
0x10 - Couple maximal, en pour-mille (0,1%), 2 octets de 0 à 1000
0x12 - Phase - la doc dit "fonction spéciale"
0x13 - Activation/désactivation des protections (surcharge, surchauffe, surintensité, plage de tension)
0x14 - Activation/désactivation des protections, visualisation par LED ??? - Comme si une LED pouvait s'allumer en cas d'erreur
0x15 - Coefficient P du correcteur de position
0x16 - Coefficient D du correcteur de position
0x17 - Coefficient I du correcteur de position
0x18 - Force de démarrage minimale (en 0,1% de la force maximale)
0x19 - Saturation du terme intégral du correcteur de position
0x1A - Zone "morte" dans le sens horaire (en nombre de pas codeur)
0x1B - Zone "morte" dans le sens anti-horaire (en nombre de pas codeur)
0x1C - Limite de courant (par pas de 6,5 mA - 2 octets). Valeur de 0 à 500, ce qui correspond à un courant maximal de 3,25 A
0x1E - Résolution angulaire (de 1 à 3) - un moyen d'obtenir du multitour au détriment de la précision angulaire. Attention, lire la doc !
0x1F - Offset de position - le bit 11 indique le signe de l'offset (2 octets)
0x21 - Mode de fonctionnement - 0 : Servomoteur, 1 : Vitesse constante, 2 : Vitesse régulée, 3 : mode pas à pas. Nous reviendrons plus tard sur ces différents modes.
0x22 - Couple de protection - couple fourni après que le servomoteur soit entré en protection à cause d'une surcharge du couple, en 1%.
0x23 - Temps pendant lequel la protection de couple reste active après s'être déclenchée - en 10 ms.
0x24 - Couple déclenchant la protection - en 1%
0x25 - Coefficient P du correcteur de vitesse
0x26 - Temps pendant lequel la protection contre les sur-intensités reste active en 10 ms.
0x27 - Coefficient I du correcteur de vitesse

SRAM en lecture et écriture

Adresse - Fonction
0x28 - Active ou désactive le couple (la commande) du servomoteur
0x29 - Consigne d'accélération, en 100 pas/s²
0x2A - Consigne de position, de -32766 à 32766, en pas
0x2C - __Je ne me prononce pas__
0x2E - Consigne de vitesse, de -32766 à 32766, en pas/s
0x30 - Limite de couple. Valeur initialisée à celle indiquée au registre 0x10 au démarrage du servo. peut être ajustée ici.
0x31 à 0x36 - registres réservés (ne pas toucher)
0x37 - Protection de l'EPROM. à mettre à 0 pour modifier l'EPROM, à remettre à 1 juste après pour la re-protéger.

SRAM en lecture seule

Adresse - Fonction
0x38 - Position actuelle (2 octets)
0x3A - Vitesse actuelle (2 octets)

0x3F - Température actuelle
0x40 - État "écriture tamporisée"
0x41 - État du servo. 0 pas d'erreur, sinon voir la doc.
0x42 - Indicateur de mouvement. 1 : en mouvement, 0 : à l'arrêt
0x43 & 0x44 - Registres réservés (ne pas toucher)
0x45 - Courant consommé

Avec ceci vous une bonne idée des capacités des servomoteurs. Si une bonne bibliothèque doit vous donnez les fonctions adéquates, savoir comment marche le servomoteur est important pour bien l'utiliser. Et parfois, la bibliothèque est un peu "jeune" ...

Avec les deux fonctions ci-dessous, vous devriez pouvoir lister les registres de votre servomoteur :

```
int lire_registre(int servo_id, int registre_adresse){
  return sms_sts.readByte(servo_id, registre_adresse);
}

void lire_tous_les_registres(int servo_id){
  char message[200]="";
  for (int i=0; i<0x46; i++){
    sprintf(message, "registre 0x%x: %d\n", i, lire_registre(servo_id, i));
    Serial.print(message);
  }
}
```

Le prochain épisode sera consacré aux différents modes du servomoteur.

Keuronde

Posté hier, 21:31

Les différents modes

Moteur à courant continue (Mode 2)

Dans ce mode, vous définissez une tension à envoyer au moteur. Et le moteur tourne.
C'est une boucle ouverte simple. Le retour du codeur n'est pas utilisé par le servomoteur.

L'intérêt dans un projet me parait assez faible, vu les autres modes disponibles.

L'initialisation se fait ainsi :

```
sms_sts.writeByte(SERVO_ID, SMS_STS_MODE, 2); // SMS_STS_MODE vau
```

La commande doit être comprise entre 1000 et -1000.

- Les bit 0 à 9 contiennent la valeur de la vitesse
- Le bit 10 son signe (0 anti-horaire, 1 horaire)

Voici le code pour l'utiliser :

```
// Construction de la commande
commande_moteur_tmp = commande_moteur;
```

```
  neg = 1;
  commande_moteur_tmp = -commande_moteur_tmp;
 }

 // Envoi de la commande
 sms_sts.writeWord(SERVO_ID, 0x2C, commande_moteur_tmp | neg << 10);
```

Note: J'ai des soucis dès que la consigne dépasse les 500. Mais je ne vais pas creuser plus...

Moteur asservi en vitesse (Mode 1)

Dans ce mode, vous indiquez une consigne de vitesse (un nombre de pas par seconde) que le moteur doit atteindre. Le servomoteur dispose d'un correcteur PI que vous pouvez ajuster en fonction de votre application.

L'initialisation se fait ainsi :

```
sms_sts.writeByte(SERVO_ID, SMS_STS_MODE, 1); // SMS_STS_MODE vau
```

ou

```
sms_sts.WheelMode(SERVO_ID);
```

L'envoi de la consigne se fait en écrivant les registres 0x2E et 0X2F.
Les 15 premiers bits contiennent la valeur absolue, le 16e le signe.

Soit vous utilisez votre fonction :

```
vitesse_tmp = vitesse;
neg = 0;
if(vitesse_tmp < 0){
  vitesse_tmp = -vitesse_tmp;
  neg = 1;
}
sms_sts.writeWord(SERVO_ID, 0x2E, vitesse_tmp | neg << 15);
```

Soit vous utilisez la fonction de la bibliothèque

```
sms_sts.WheelMode(SERVO_ID);
```

C'est, comme la fonction de la bibliothèque l'indique, le mode idéal pour piloter des roues. C'est aussi un mode intéressant pour travailler le réglage d'un asservissement.

Sur l'image ci-dessous, vous voyez la consigne et la vitesse réelle, ainsi que la commande moteur. Sur deux cycles, j'ai freiné - légèrement - le servomoteur. Vous observé que la commande monte en flèche et que la vitesse se stabilise tout près de la consigne malgré la perturbation.

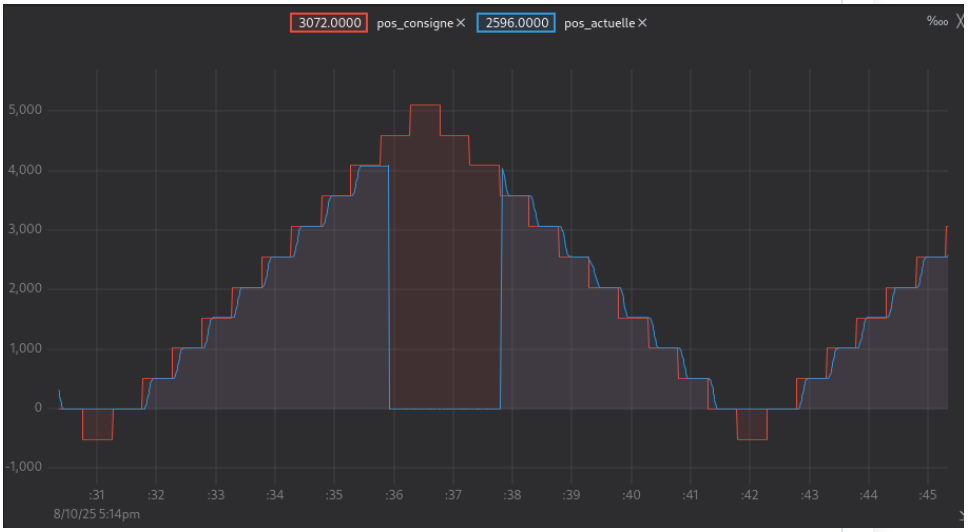
Servo mode (0)

Nous nous rapprochons beaucoup du servomoteur classique avec une consigne de position.

Simple tour

Le principe est relativement simple, vous envoyer une position, et le servomoteur y va.

Avec la configuration par défaut, en faisant varier la consigne de position de -512 à 5120, nous observons des butées logicielles à 0 et à 4093. Notons aussi que la position renvoyée peut reboucler à 0 comme indiqué sur le graphique. Nous avons globalement le fonctionnement d'un servomoteur classique, sauf que nous avons le retour de position et que nous pouvons régler également la vitesse.



Multi-tour

Ces servomoteurs n'ont pas de butées mécaniques, et nous avons vu qu'ils pouvaient réaliser plusieurs tours.
En mettant à 0 la butée minimale (2 octets) et la butée maximale (2 octets),

mécanismes avec démultiplication. Ce mode présente deux inconvénients :

- Au bout d'un certain nombre de tour, le compteur reboucle. Sur le modèle que j'ai, je peux faire 8 tours dans chaque direction à partir du zéro. Si je dépasse cette limite, le servomoteur part en sens inverse et réalise les 16 tours.
- Si le servomoteur est dé-alimenté, il perd le nombre de tour qu'il a effectué et se réinitialise dans la plage du premier tour.

Par défaut, le servomoteur renvoie une position qui est toujours comprise dans le premier tour, entre 0 et 4096. Il y a un bit du registre Phase à modifier pour obtenir la position entre -8 et +8 tours. Mais activer ce bit ne réglera pas les deux inconvénients précédents.

Basse résolution

Il y a un paramètre qui permet de ne compter qu'un pas sur 2 ou un pas sur 3. Ceci dégrade la précision du codeur mais permet de multiplier par 3 (au max) la plage sur laquelle le servomoteur peut être contrôlé en position. Ceci n'a d'intérêt qu'en multi-tour. J'ai noté un comportement bizarre des butées logicielles et un écart x2 ou x3 entre la consigne et la position réelle.

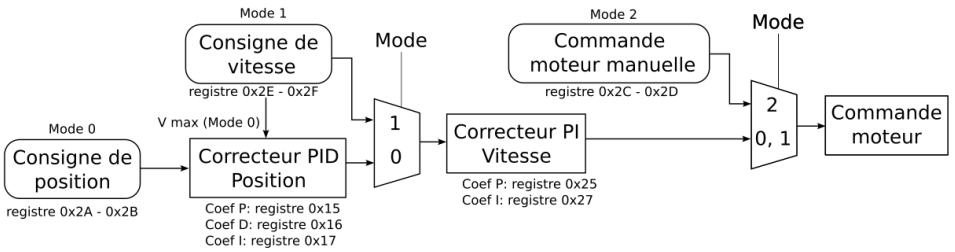
Pour activer le mode basse résolution :

```
sms_sts.writeByte(SERVO_ID, 0x1E, 3);
```

Passer à 1 le bit 4 du registre "Phase" règle les comportements bizarre (au moins l'écart entre la consigne et la position réelle).
Attention, la valeur par défaut pourrait changer en fonction des modèles et version des servomoteurs.

```
sms_sts.writeByte(SERVO_ID, 0x12, 0x7C);   // Registre de "Phase", valeur pa
```

Un schéma pour comprendre comment tout ça s'emboîte !



Un mot sur la bibliothèque Feetech

Le fabriquant fournis une bibliothèque qui permet d'accéder facilement aux fonctions des bases. Robot-Maker fourni une évolution de la bibliothèque un peu plus claire. Voici en vrac les points qui ont pu m'interroger ou qui mérite une précision...

La fonction Feedback

PWM, Tension, température, erreur, indicateur de mouvement et courant consommé) d'un servomoteur en une seule transaction de communication. par la suite, appeler les fonctions de lecture d'état (ReadSpeed, ReadPos, etc...) iront consulter ce tampon si l'identifiant passé en paramètre vaut -1. C'est ce qui est fait dans le code d'exemple, mais sans explications.

Les fonctions readByte et writeByte

Les fonctions readByte et writeByte permettent de lire et modifier les registres et vous permettent d'accéder à l'ensemble des fonctionnalités des servomoteurs. Les fonctions readWord et writeWord permettent d'écrire des valeurs de 16 bits sur deux registres consécutifs, en ne donnant que l'adresse du premier.

Avec ces fonctions vous pouvez recréer toute la bibliothèque et créer les fonctions qui vous manquent.

Les adresses des registres

Le fichier .h fourni quelques adresses de registre mais pas tous, loin de là. c'est dommage.

Le multi-tour

Le moyen d'activer le multi-tour n'est pas indiqué dans la bibliothèque. Les limitations du multi-tour ne sont détaillées nulle part, c'est dommage.

Modération du sujet · Retourner dans Demandes d'informations sur les produits de la boutique

Bienvenue sur Robot Maker → News de la boutique Robot Maker → Demandes d'informations sur les produits de la boutique →
Présentation et fonctionnement des servomoteurs FEETECH
Débuté par EVRIS, 19 fÃ©vr. 2025
Feetech, Servomoteur

0 réponses
1â€¯886 vues
EVRIS
19 fÃ©vr. 2025

Sujets généraux → Electronique →
Recherche de pilote pour carte WIT Motion 16 servomoteurs
Débuté par Gecko64, 07 mars 2023
wit motion, servomoteur

2 réponses
1â€¯103 vues
Gecko64
07 mars 2023

Procédés de fabrications, techniques et choix de matériels et d'outillages →
Impression 3D et Imprimantes 3D →
[WIP] Main humaine Kwawu motorisé en 3D
Débuté par Robot Urbie en légo, 10 avril 2020
Kwawu, Servomoteur, Hitec

10 réponses
4â€¯400 vues
Oracid
27 dÃ©c. 2021

Lego - Arduino Kame Quadruped
Débuté par Oracid, 26 nov. 2018
Quadrupède, Lego, HC05, Oracid et 2 de plus... 1 2

POPULAIRE 28 réponses
11â€¯041 vues

Oracid
09 janv. 2019

Découvertes → Présentations de produits robotique → Servomoteurs, moteurs et autres actionneurs → Servomoteurs →
Les servomoteurs en vidéo.
Débuté par Oliver17, 31 aoÃ»t 2018
Servomoteur, Barbatronic, video

0 réponses
2â€¯106 vues

Oliver17
31 aoÃ»t 2018