Aabu

Zeste de savoir

Piloter un servomoteur sur une pyboard avec MicroPython

14 décembre 2022

Table des matières

Introduction

La pyboard permet de facilement piloter des servomoteurs. Ce billet résume comment utiliser des servomoteurs avec une pyboard et en particulier comment les calibrer s'il est nécessaire de les piloter avec une certaine précision.

1. Connecter le servomoteur

Un petit servomoteur tel qu'on les utilise pour des projets personnels se connectent avec un connecteur disposant de trois fils:

— un fil de masse;
— un fil avec la tension d'alimentation (environ 5V);
— un dernier fil pour la commande.

La pyboard prévoit quatre emplacements sur la carte, avec les pins adéquates côte à côte prête à connecter le connecteur. Cela permet donc de juste brancher le connecteur du servomoteur et connecter ainsi la masse à la masse de la carte, l'alimentation à une alimentation sur la carte et la commande à un ADC qui s'occupera de fournir une commande adéquate.

Il faut prendre soin de bien mettre le connecteur dans le bon sens, sinon vous risquez d'inverser la masse et la commande, ce qui ne fonctionnera pas.

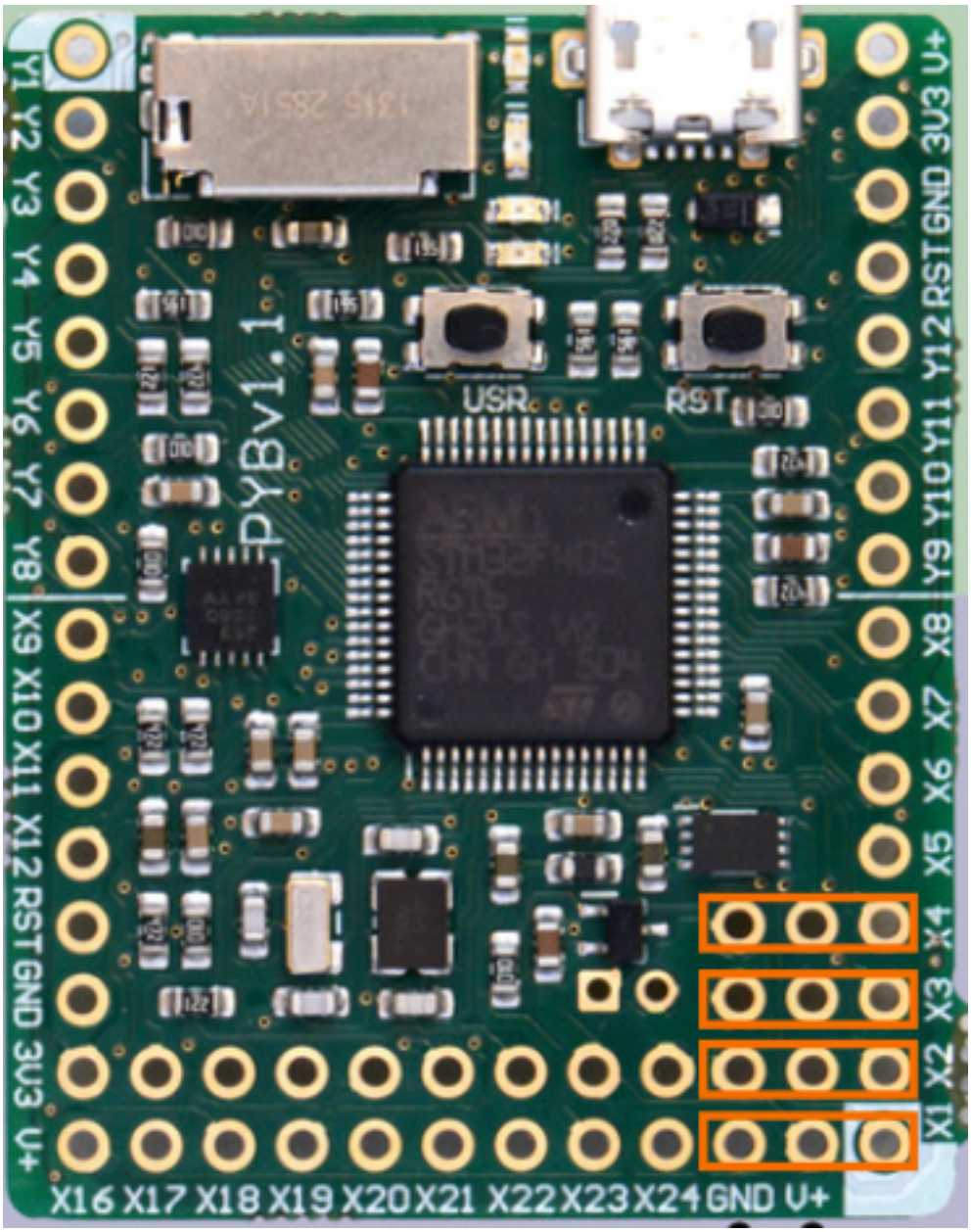

FIGURE 1.1. – Encadrés en orange, les quatre emplacements disponibles sur la carte pour piloter des servomoteurs.

2. Piloter le servomoteur avec l'objet Servo

La bibliothèque MicroPython de la pyboard contient un objet dédié au pilotage de servomoteurs qui facilite grandement le travail.

2.1. Importer le module `pyb`

Il faut d'abord s'assurer de bien importer le module `pyb` qui contient la classe `Servo`.

```
>>> import pyb
```

2.2. Créer un objet de type `Servo`

Ensuite, on crée un objet `Servo` qui servira à piloter le servomoteur. Le constructeur `Servo` prend un entier en argument, qui correspond à l'emplacement qu'on souhaite piloter (1 pour X1, 2 pour X2, etc.). Dans mon cas, ça sera l'emplacement 2.

```
>>> servo = pyb.Servo(2)
```

2.3. Contrôler l'angle

Il suffit ensuite d'utiliser la méthode `angle()` pour contrôler l'angle. La méthode prend en argument un angle *en degrés*.

```
>>> servo.angle(30)  # se positionner 30 degrés dans un sens
>>> servo.angle(-30)  # se positionner 30 degrés dans l'autre sens
>>> servo.angle(0)  # revenir à la position centrale
```

On peut aussi avoir envie de réaliser ce mouvement en temps limité, il existe pour ça un argument optionnel:

```
>>> servo.angle(30, 1000)  # se positionner 30 degrés dans un
    sens, et le faire en 1000 ms
```

Quand on ne spécifie pas `time`, le servomoteur essaiera de rejoindre la position indiquée le plus vite possible.

Vous aurez très probablement remarqué que l'angle que vous donnez ne correspond pas à l'angle réel, c'est pour ça qu'il faut calibrer le servomoteur.

2.4. Contrôler la vitesse

Avec certains types de servomoteurs, c'est la vitesse qu'on contrôle plutôt que l'angle. Dans ce cas, il faut utiliser la méthode `speed()`. Je ne m'étendrai pas plus dans ce billet, ce n'est pas le type de servomoteur dont je me sers.

3. Calibrer le servomoteur

3.1. Pourquoi calibrer

Un servomoteur est fait pour se positionner à un angle physique correspondant à une certaine commande. Cependant, le servomoteur n'est pas commandé par un angle directement, mais par des impulsions. Quand on utilise la méthode `angle()`, l'objet effectue une conversion entre l'angle et la largeur des impulsions à émettre.

La manière de prendre en compte ces impulsions peuvent varier d'un modèle de servomoteur à un autre, voire de deux servomoteurs d'un même modèle. Pour avoir un pilotage précis, il vaut mieux que l'angle de la commande soit exactement le même que celui auquel ira réellement le servomoteur. Pour ce faire, il faut paramétrer la conversion de l'angle en impulsions pour que l'angle réel et la commande soient bien en accord, c'est ça qu'on appelle calibration.

3.2. Paramètres de calibration

La calibration d'un objet `Servo` se fait à l'aide de six paramètres :

— la durée minimale de pulsation (en millisecondes);
— la durée maximale de pulsation (en millisecondes);
— la pulsation pour un angle de 0° (en millisecondes);
— la pulsation pour un angle de 90° (en millisecondes);
— (inutile dans mon cas) la pulsation correspondant à la vitesse max (en millisecondes).

On peut obtenir la calibration actuelle avec la méthode `calibration()` de l'objet `Servo`, appelée sans arguments.

```
>>> servo.calibration()
(640, 2420, 1500, 2470, 2200)
```

Les six entiers sont les paramètres listés ci-dessous, dans le même ordre.

On peut donner de nouveaux paramètres de calibration en appelant `calibration()` avec les nouveaux paramètres:

```
>>> servo.calibration(380, 2290, 1310, 2170, 2170)
```

3.3. Déterminer les paramètres

Avec un peu de méthode, il est possible de déterminer les paramètres adéquats assez rapidement. Une aide très utile est la méthode `pulse_width()`, qui permet de donner une commande directement sous forme de durée d'impulsion plutôt que sous forme d'angle.

```
>>> servo.pulse_width(500)  # impulsions de 500 ms
```

Ensuite, je procède de la manière suivante:

1. Descendre progressivement vers 0 pour trouver la valeur minimale d'impulsion. Quand le servomoteur se met à vibrer (à l'oreille), c'est que je suis en butée et que l'impulsion est bien la durée minimum.
2. Augmenter progressivement la durée d'impulsion pour trouver la valeur maximum. Quand le servomoteur se met à vibrer, c'est qu'on a atteint la butée et que l'impulsion est bien la durée maximum.
3. Trouver la durée d'impulsion pour le centre (0°) par tâtonnement, en prenant soin de prendre un bon point de repère sur l'axe du servomoteur. Les servomoteurs sont en général plus ou moins symétriques, donc il faut chercher à mi-chemin entre l'impulsion minimale et maximale.
4. Trouver la durée d'impulsion pour 90° en procédant de manière similaire au zéro. Pour un servomoteur avec 180° de débattement, la durée d'impulsion pour 90° est proche de la durée maximale.

Les étapes 3 et 4 peuvent être un peu délicate, voire longues si vous voulez être très précis ou si vous avez du mal à garder un repère.

3.4. Enregistrer les paramètres et vérifier la bonne calibration

Une fois vos paramètres déterminés, *notez-les quelque part* pour référence future. Vous pouvez alors mettre à jour l'objet `Servo` avec vos paramètres:

```
>>> servo.calibration(380, 2290, 1310, 2170, 2170)
```

Le dernier paramètres sert pour le pilotage en vitesse. Je recopie juste la valeur d'avant vu que je n'en ai pas l'usage avec mes servomoteurs.

On s'assure enfin que tout est bon en vérifiant le bon positionnement physique du servomoteur en réponse à des commandes d'angles faciles à vérifier:

```
>>> servo.angle(90)
>>> servo.angle(45)
>>> servo.angle(0)
>>> servo.angle(-90)
>>> servo.angle(-45)
```

Si ça ne tombe pas juste, vous êtes bons pour corriger votre calibration!

Conclusion

Vous pouvez retrouver tous les détails nécessaires sur la classe Servo dans la documentation du module pyb . La documentation officielle propose aussi un petit tuto sur le pilotage de servomoteurs .

Miniature du billet: logo de MicroPython (source).