Servomoteurs et bibliothèque Servo pour ESP32

Houssem-eddine LAHMER

4 mai 2025

Plan du cours

Qu'est-ce qu'un servomoteur ?

- Un **servomoteur** est un moteur capable de maintenir une position angulaire précise
- Composants principaux :
  - Moteur DC
  - Système d'engrenages réducteurs
  - Circuit de contrôle avec retour de position
  - Potentiomètre de retour de position
- Utilisations courantes :
  - Robotique
  - Modélisme (avions, voitures RC)
  - Automatismes
  - Projets IoT

Types de servomoteurs

Servo standard (180°)

- Rotation limitée entre 0° et 180°
- Précision angulaire élevée
- Exemple : SG90, MG996R

Servo à rotation continue

- Rotation complète à 360°
- Contrôle de vitesse au lieu de position
- Utile pour les roues, convoyeurs

Caractéristiques techniques importantes :

- Couple (torque) - en kg·cm ou oz·in
- Vitesse de rotation
- Tension d'alimentation (typiquement 4.8V-6V)
- Poids et dimensions

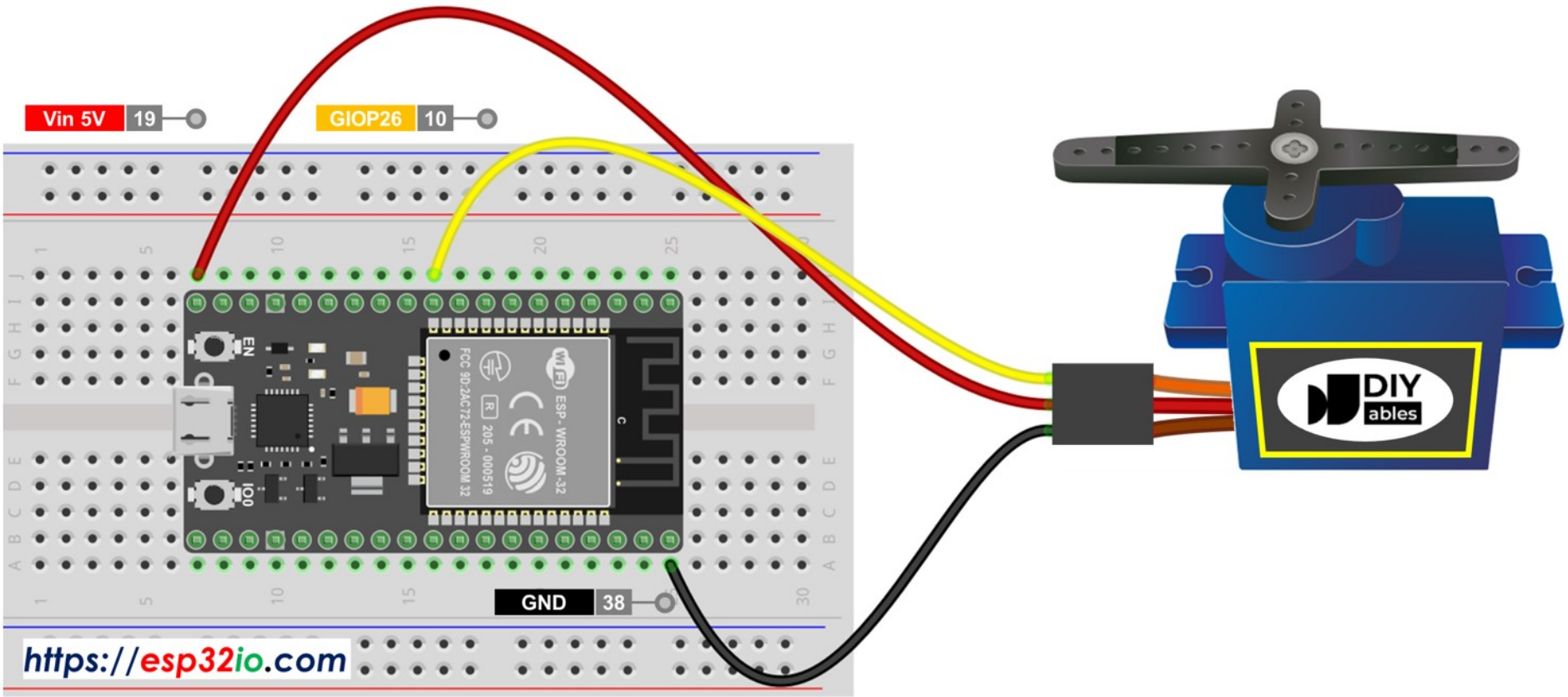
Principe de fonctionnement

- Contrôle par **signaux PWM** (Pulse Width Modulation)
- Largeur d'impulsion entre 1ms et 2ms dans une période de 20ms
  - 1ms = position 0° (minimum)
  - 1.5ms = position 90° (centre)
  - 2ms = position 180° (maximum)
- Circuit interne :
  - Compare la position demandée (signal PWM)
  - Avec la position réelle (potentiomètre)
  - Ajuste le moteur jusqu'à concordance

L'ESP32 et ses capacités PWM

- L'ESP32 dispose de 16 canaux PWM indépendants
- Résolution configurable jusqu'à 16 bits
- Fréquence réglable jusqu'à 40MHz
- Idéal pour contrôler plusieurs servomoteurs simultanément
- Avantages par rapport à d'autres microcontrôleurs :
  - Plus de canaux PWM
  - Meilleure résolution et précision
  - Processeur double cœur à 240MHz
  - WiFi et Bluetooth intégrés

Connexion d'un servomoteur à l'ESP32



Connexion d'un servomoteur à l'ESP32

Branchement :

- Fil rouge → 5V (ou 3.3V pour certains modèles)
- Fil marron/noir → GND
- Fil orange/jaune → GPIO de l'ESP32

Considérations d'alimentation :

- Alimentation séparée recommandée pour plusieurs servos
- Condensateur 100-470F en parallèle peut aider à stabiliser
- Ne pas alimenter des servos puissants directement par l'ESP32

La bibliothèque ESP32Servo

- Extension de la bibliothèque Servo standard d'Arduino
- Spécifiquement adaptée pour l'ESP32
- Fonctionnalités :
  - Contrôle jusqu'à 16 servomoteurs simultanément
  - Compatible avec tous les GPIOs de l'ESP32
  - Gestion automatique des timers matériels
  - Support pour servos standard et à rotation continue
- Installation via le gestionnaire de bibliothèques Arduino :
  - Rechercher "ESP32Servo" par Kevin Harrington
  - Ou via GitHub : `https://github.com/madhephaestus/ESP32Servo`

Fonctions principales de la bibliothèque

```
// Cr ation d'un objet servo
Servo monServo;

// Attacher le servo    une broche GPIO
monServo.attach(pin);  // Version simple
monServo.attach(pin, min, max);  // Avec impulsions min/max

// Contr le de position (0-180 degr s)
monServo.write(angle);  // 0    180

// Contr le direct par dur e d'impulsion
monServo.writeMicroseconds(us);  // 1000    2000

// Lire la position actuelle
int position = monServo.read();

// D tacher le servo (lib re le canal PWM)
monServo.detach();
```

Création et attachement d'un servo

```
1 // Cr ation d'un objet servo
2 Servo monServo;
3
4 // Attacher le servo    une broche GPIO (version simple)
5 monServo.attach(pin);
6
7 // Attacher avec configuration  d impulsions  min/max
8 monServo.attach(pin, min, max);
```

Contrôle de position par angle

```
1 // Positionnement en degr s (0     180)
2 monServo.write(angle);
```

Contrôle direct par durée d'impulsion

```
1 // Sp cification de la largeur d impulsion en
    microsecondes
2 monServo.writeMicroseconds(us); // typiquement 1000 2000
      s
```

Lecture de la position actuelle

```
1 // Retourne la dernière position écrite (0 à 180)
2 int position = monServo.read();
```

Détacher le servo

```
1 // Libère le canal PWM, arrête des impulsions
2 monServo.detach();
```

Exemple de code simple

```
#include <ESP32Servo.h>

Servo monServo;  // Cr ation de l'objet servo
int servoPin = 13;  // Broche GPIO    laquelle le servo est
    connect

void setup() {
  // Attache le servo au GPIO sp cifi
  monServo.attach(servoPin);

  // Position initiale au centre
  monServo.write(90);

  delay(1000);  // Attendre que le servo atteigne sa
    position
}

void loop() {
  // Balayage de 0    180 degr s
  for(int angle = 0; angle <= 180; angle += 5) {
    monServo.write(angle);
```

Initialisation du servo

```
#include <ESP32Servo.h>

Servo monServo;          // Cr ation de l'objet servo
int servoPin = 13;       // Broche GPIO du servo

void setup() {
  // Attache le servo au GPIO sp cifi
  monServo.attach(servoPin);

  // Position initiale au centre (90  )
  monServo.write(90);

  delay(1000);  // Laisser le temps au servo  d atteindre
    la position
}
```

Boucle de balayage 0–180°

```
void loop() {
  // Balayage de 0    180 degr s
  for(int angle = 0; angle <= 180; angle += 5) {
    monServo.write(angle);
    delay(50);
  }

  // Balayage de 180    0 degr s
  for(int angle = 180; angle >= 0; angle -= 5) {
    monServo.write(angle);
    delay(50);
  }
}
```

Déclaration des objets Servo et des broches

```
#include <ESP32Servo.h>

// Cr ation de multiples objets servo
Servo servo1;
Servo servo2;
Servo servo3;

// D finition des broches
int servo1Pin = 13;
int servo2Pin = 12;
int servo3Pin = 14;
```

Configuration initiale des servomoteurs

```
1  void setup() {
2    // Attache des servos aux broches sp cifi es
3    servo1.attach(servo1Pin);
4    servo2.attach(servo2Pin);
5    servo3.attach(servo3Pin);
6
7    // Position initiale    90 degr s pour chaque servo
8    servo1.write(90);
9    servo2.write(90);
10   servo3.write(90);
11
12   delay(1000); // Attente pour stabiliser les servos
13 }
```

Boucle de balayage des servomoteurs

```
void loop() {
  // Balayage de 0    180 degr s
  for(int angle = 0; angle <= 180; angle += 5) {
    servo1.write(angle);
    servo2.write(180 - angle); // Mouvement inverse
    servo3.write(angle / 2);   // Demi-amplitude
    delay(30);
  }

  delay(500); // Pause entre les balayages

  // Balayage de 180    0 degr s
  for(int angle = 180; angle >= 0; angle -= 5) {
    servo1.write(angle);
    servo2.write(180 - angle);
    servo3.write(angle / 2);
    delay(30);
  }

  delay(500); // Pause entre les balayages
}
```

Considérations supplémentaires

- **Alimentation des servomoteurs :** Il est recommandé d'utiliser une alimentation externe pour les servomoteurs, surtout si vous en contrôlez plusieurs, afin d'éviter de surcharger l'ESP32.
- **Choix des broches GPIO :** Assurez-vous que les broches utilisées sont compatibles avec la génération de signaux PWM nécessaires pour contrôler les servomoteurs.
- **Limitations matérielles :** Certains modèles d'ESP32 peuvent avoir des limitations sur le nombre de servomoteurs contrôlables simultanément. Consultez la documentation de votre carte pour plus de détails.

Introduction à Wokwi

- Wokwi est un simulateur en ligne pour Arduino, ESP32 et autres microcontrôleurs
- Avantages pour les tests de servomoteurs :
  - Pas besoin de matériel physique
  - Visualisation immédiate du comportement
  - Pas de risques de dommages matériels
  - Facilité de partage des projets
  - Intégration avec l'IDE Arduino
- Accès : `https://wokwi.com/`

Configuration d'un projet servo sur Wokwi

Étapes pour créer un projet ESP32 avec servo :

1. Créer un nouveau projet ESP32
2. Ajouter un composant "Servo Motor" depuis le panneau de composants
3. Connecter les broches du servo :
   - Signal $\rightarrow$ GPIO de l'ESP32 (ex : GPIO13)
   - $+ \rightarrow$ 5V
   - - $\rightarrow$ GND
4. Écrire le code (utilisant la bibliothèque ESP32Servo)
5. Démarrer la simulation avec le bouton "Start"

Configuration du fichier diagram.json :

- Définit les composants et les connexions
- Permet de spécifier les propriétés du servo
- Peut être édité directement pour des configurations personnalisées

Exemple complet : balayage servo sur Wokwi

```
// Fichier sketch.ino
#include <ESP32Servo.h>

Servo myservo;
int servoPin = 13;

void setup() {
  Serial.begin(115200);
  myservo.attach(servoPin);
  Serial.println("ESP32 Servo Test");
}

void loop() {
  // Balayage de 0    180
  for (int pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos);
    Serial.print("Position: ");
    Serial.println(pos);
    delay(15);
  }
```

Configuration Wokwi : diagram.json

```
{
  "version": 1,
  "author": "Votre Nom",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0, "attrs": {} },
    { "type": "wokwi-servo", "id": "servo1", "top": 100, "left": 100, "attrs": {} }
  ],
  "connections": [
    [ "esp:GND.1", "servo1:GND", "black", [ "v0" ] ],
    [ "esp:5V", "servo1:V+", "red", [ "v0" ] ],
    [ "esp:D13", "servo1:SIG", "orange", [ "v0" ] ]
  ],
  "dependencies": {}
}
```

Déboggage sur Wokwi

Outils de débogage disponibles :

- Console série (moniteur)
- Visualisation du signal PWM
- Inspection des broches GPIO
- Observation du mouvement du servo en temps réel
- Ajustement de la vitesse de simulation

Problèmes courants et solutions :

- Servo ne bouge pas :
  - Vérifier les connexions
  - Vérifier le numéro de GPIO utilisé
  - S'assurer que la bibliothèque est correctement importée
- Mouvements erratiques :
  - Vérifier les valeurs min/max dans attach()
  - Ajuster la temporisation entre les mouvements
- Consommation CPU élevée :
  - Réduire la fréquence des mises à jour du servo

Projets utilisant des servomoteurs avec ESP32

Projets robotiques :

- Bras robotique
- Robot marcheur (hexapode, quadrupède)
- Système de pan-tilt pour caméra

Automatisation domestique :

- Contrôle de store/volet roulant
- Serrure connectée
- Distribution automatique de nourriture pour animaux

Projets IoT :

- Station météo avec anémomètre
- Contrôle à distance via WiFi/Bluetooth
- Système de suivi solaire pour panneaux photovoltaïques

Projet : contrôle de servo par potentiomètre

```
#include <ESP32Servo.h>

Servo myservo;
int servoPin = 13;
int potPin = 34;  // Broche de lecture analogique

int potValue;     // Valeur du potentiom tre (0-4095)
int angle;        // Angle du servo (0-180)

void setup() {
  Serial.begin(115200);
  myservo.attach(servoPin);
}

void loop() {
  // Lecture du potentiom tre
  potValue = analogRead(potPin);

  // Conversion de la valeur (0-4095) vers angle (0-180)
  angle = map(potValue, 0, 4095, 0, 180);
```

Projet : contrôle WiFi de servomoteurs

```
#include <WiFi.h>
#include <WebServer.h>
#include <ESP32Servo.h>

// Configuration WiFi
const char* ssid = "VotreSSID";
const char* password = "VotreMotDePasse";

// Cr ation du serveur web sur le port 80
WebServer server(80);

// Configuration des servos
Servo servo1;
Servo servo2;
int servo1Pin = 13;
int servo2Pin = 12;
int servo1Pos = 90;  // Position initiale
int servo2Pos = 90;  // Position initiale

void setup() {
  Serial.begin(115200);

  // Connexion des servos
  servo1.attach(servo1Pin);
  servo2.attach(servo2Pin);

  // Positionnement initial
  servo1.write(servo1Pos);
  servo2.write(servo2Pos);

  // Connexion au WiFi
  WiFi.begin(ssid, password);
  Serial.print("Connexion au WiFi");
```

Bonnes pratiques et optimisations

Optimisations hardware :

- Alimentation adéquate et séparée pour les servos puissants
- Découplage avec condensateurs pour réduire le bruit
- Isolation du bruit entre les circuits de puissance et de signal

Optimisations software :

- Utilisation de la programmation orientée objet pour gérer plusieurs servos
- Éviter les délais bloquants avec millis() ou FreeRTOS
- Calibration par servo pour compenser les variations de fabrication
- Lissage des mouvements avec des fonctions d'accélération/décélération

Considérations de sécurité :

- Limites logicielles pour éviter les dommages mécaniques
- Gestion des positions "sûres" en cas de perte de connexion
- Vérification des commandes entrantes avant exécution

Récapitulatif

- Les servomoteurs permettent un contrôle précis de la position angulaire
- L'ESP32 offre d'excellentes capacités pour contrôler les servomoteurs :
  - 16 canaux PWM indépendants
  - Contrôleur performant pour la logique
  - Connectivité WiFi/BT pour contrôle distant
- La bibliothèque ESP32Servo simplifie grandement la programmation
- Wokwi permet de prototyper et tester sans matériel physique
- Applications diverses : robotique, domotique, IoT

Ressources complémentaires

- Documentation officielle :
  - ESP32Servo : `https://github.com/madhephaestus/ESP32Servo`
  - ESP32 : `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/`
  - Wokwi : `https://docs.wokwi.com/`
- Tutoriels recommandés :
  - Random Nerd Tutorials : ESP32 avec servomoteurs
  - Electropeak : Guide complet sur les servomoteurs
  - DroneBot Workshop : Utilisation avancée des servos
- Livres :
  - "ESP32 Programming for the Internet of Things" par Agus Kurniawan
  - "Robotics with the ESP32" par Manoj R. Thakur

Merci pour votre attention !

Des questions ?