**Πείραμα με σερβοκινητήρα 2-αξόνων(2 σερβοκινητήρες):**

1. **Περιγραφή:**

Στο τρέχον πείραμα με τη βοήθεια του Raspberry Pi ,μαζί με τον ανάλογο κώδικα σε Python, ελέγχω σερβοκινητήρα δύο αξόνων(Χ,Υ).

1. **Υλικά:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **1\* Raspberry pi** | **1\* GPIO Extension board** | **1\* 40 pin colorful jumper wires** |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **1\* Breadboard** | **Jumper Wires** | **1\* two-axis servo with 0.1s/60 degree operating speed** |

1. **Γνώση υλικού:**

Ένας σερβοκινητήρας δύο αξόνων ή σερβοκινητήρας δύο βαθμών ελευθερίας, είναι ένας τύπος σερβοκινητήρα που μπορεί να ελέγχει την κίνηση σε δύο διαφορετικές κατευθύνσεις ή άξονες ταυτόχρονα. Αυτοί οι κινητήρες χρησιμοποιούνται συνήθως στη ρομποτική, στα συστήματα αυτοματισμού και σε άλλες εφαρμογές όπου απαιτείται ακριβής και συντονισμένη κίνηση.Στην ουσία έχω δύο σερβοκινητήρες όπου ο καθένας είναι υπεύθυνος για τον αντίστοιχο άξονα.

**Λειτουργικότητα:**

Έλεγχος διπλού άξονα:

Οι σερβοκινητήρες δύο αξόνων έχουν σχεδιαστεί για να παρέχουν έλεγχο της κίνησης σε δύο κάθετους άξονες, που συνήθως αναφέρονται ως Χ και Υ. Αυτό σημαίνει ότι μπορούν να κινούνται τόσο σε οριζόντια όσο και σε κάθετη κατεύθυνση.

Ακριβής τοποθέτηση:

Οι σερβοκινητήρες είναι γνωστοί για την ακρίβειά τους στον έλεγχο της θέσης. Οι σερβοκινητήρες δύο αξόνων προσφέρουν ακριβή έλεγχο της θέσης του άξονα του κινητήρα και στους δύο άξονες. Αυτή η ακρίβεια επιτυγχάνεται μέσω μηχανισμών ανατροφοδότησης, όπως οι κωδικοποιητές, οι οποίοι παρέχουν συνεχώς πληροφορίες σχετικά με την τρέχουσα θέση του κινητήρα.

Έλεγχος ταχύτητας:

Αυτοί οι σερβοκινητήρες επιτρέπουν τον ακριβή έλεγχο της ταχύτητας με την οποία κινείται ο κινητήρας και στους δύο άξονες. Αυτό είναι ιδιαίτερα χρήσιμο για εφαρμογές όπου απαιτείται ομαλή και ελεγχόμενη κίνηση.

Επιτάχυνση και επιβράδυνση:

Τα συστήματα σερβοκινητήρων μπορούν να ελέγχουν την επιτάχυνση και την επιβράδυνση του κινητήρα, διασφαλίζοντας ότι η κίνηση ξεκινά και σταματά ομαλά. Αυτό το χαρακτηριστικό είναι ζωτικής σημασίας για την αποφυγή ξαφνικών τραντάγματος και την προστασία των μηχανημάτων.

Έλεγχος ανατροφοδότησης:

Τα περισσότερα συστήματα σερβομηχανισμού δύο αξόνων περιλαμβάνουν βρόχους ελέγχου ανατροφοδότησης που παρακολουθούν συνεχώς τη θέση του κινητήρα και πραγματοποιούν ρυθμίσεις για τη διατήρηση της επιθυμητής θέσης. Αυτός ο μηχανισμός ανατροφοδότησης διασφαλίζει ότι ο κινητήρας ακολουθεί με ακρίβεια την καθορισμένη διαδρομή.

Συγχρονισμός:

Οι σερβομηχανισμοί δύο αξόνων μπορούν να συγχρονιστούν για να δουλεύουν μαζί, κάνοντας συντονισμένες κινήσεις. Αυτό το χαρακτηριστικό είναι επωφελές σε εφαρμογές όπως η ρομποτική, όπου πολλοί άξονες πρέπει να συνεργάζονται για την επίτευξη σύνθετων εργασιών.

Εφαρμογές:

Οι σερβοκινητήρες δύο αξόνων χρησιμοποιούνται σε ένα ευρύ φάσμα εφαρμογών, όπως μηχανές CNC, εκτυπωτές 3D, ρομποτικοί βραχίονες, gimbals καμερών και πολλά άλλα. Είναι ιδιαίτερα πολύτιμοι σε εφαρμογές που απαιτούν ακριβή έλεγχο της δισδιάστατης κίνησης.

Συνολικά, οι σερβοκινητήρες δύο αξόνων παρέχουν υψηλό επίπεδο ελέγχου και ακρίβειας, καθιστώντας τους απαραίτητα εξαρτήματα σε διάφορα αυτοματοποιημένα συστήματα και συσκευές που απαιτούν ακριβή και συγχρονισμένη κίνηση σε δύο κατευθύνσεις.

**Πειραματική σύνδεση:**

|  |  |
| --- | --- |
| The 2-axis servo module | Raspberry Pi |
| Vcc | 5V |
| Gd | GND |
| PWM1 | 17 |
| PWM2 | 18 |

1. **Πειραματικό συμπέρασμα:**

Σε αυτό το παράδειγμα, χρησιμοποιούμε σήματα PWM για τον έλεγχο των θέσεων του σερβομηχανισμού. Ρυθμίστε τις τιμές του κύκλου λειτουργίας (π.χ. 2,5 για το ένα άκρο, 7,5 για το κέντρο και 12,5 για το άλλο άκρο) για να τοποθετήσετε τα σερβομηχανήματα όπως απαιτείται για τη συγκεκριμένη εφαρμογή σας**.**

**Βεβαιωθείτε ότι έχετε εγκαταστήσει τη βιβλιοθήκη RPi.GPIO στο Raspberry Pi σας, αν δεν είναι ήδη εγκατεστημένη. Μπορείτε να το κάνετε αυτό με την ακόλουθη εντολή στη γραμμή εντολών(command line):**

**pip install RPi.GPIO**

1. **Κώδικας Python:**

import RPi.GPIO as GPIO

import time

# Set the GPIO mode

GPIO.setmode(GPIO.BCM)

# Define the GPIO pins for PWM1 and PWM2

pwm1\_pin = 17  # Replace with the actual GPIO pin number for PWM1

pwm2\_pin = 18  # Replace with the actual GPIO pin number for PWM2

# Initialize the GPIO pins

GPIO.setup(pwm1\_pin, GPIO.OUT)

GPIO.setup(pwm2\_pin, GPIO.OUT)

# Create PWM instances for both servos

pwm1 = GPIO.PWM(pwm1\_pin, 50)  # 50Hz frequency

pwm2 = GPIO.PWM(pwm2\_pin, 50)

# Start the PWM signals

pwm1.start(7.5)  # Adjust the duty cycle for servo position

pwm2.start(7.5)

try:

    while True:

        # Move the servos to predefined positions

        pwm1.ChangeDutyCycle(12.5)  # Adjust these values as needed

        pwm2.ChangeDutyCycle(2.5)

        time.sleep(2)

        pwm1.ChangeDutyCycle(2.5)

        pwm2.ChangeDutyCycle(12.5)

        time.sleep(2)

except KeyboardInterrupt:

    pass

# Clean up GPIO on script exit

pwm1.stop()

pwm2.stop()

GPIO.cleanup()

**Παραλλαγή πειράματος:**

Στην παραλλαγή μαζί με το module 2-axis servo χρησιμοποιώ και το module 5-tact switches.Όταν ο χρήστης πατάει το S3, έχουμε μισή διαδρομή για τον servo1 ενώ με το s1 έχουμε μισή διαδρομή για τον servo1 στην αντίθετη κατεύθυνση.

Όταν ο χρήστης πατάει το S0, έχουμε μισή διαδρομή για τον servo2 ενώ με το s4 έχουμε μισή διαδρομή για τον servo2 στην αντίθετη κατεύθυνση.

Όταν ο χρήστης πατάει το S2, και οι δύο (servo1,servo2) πάνε στην αρχική τους θέση .

**Πειραματική σύνδεση:**

|  |  |
| --- | --- |
| The 2-axis servo module | Raspberry Pi |
| Vcc | 5V |
| Gd | GND |
| PWM1 | 17 |
| PWM2 | 18 |

|  |  |
| --- | --- |
| The five tact switches module | Raspberry Pi |
| Vcc | 5V |
| Gd | GND |
| S0-S4 | 22 , 23 , 24 , 25 , 26 |

**Κώδικας Python:**

import RPi.GPIO as GPIO

import time

# Set the GPIO mode

GPIO.setmode(GPIO.BCM)

# Define GPIO pins for servo PWM control

pwm1\_pin = 17  # Replace with the actual GPIO pin number for servo1

pwm2\_pin = 18  # Replace with the actual GPIO pin number for servo2

# Define GPIO pins for tact switches

s0\_pin = 22

s1\_pin = 23

s2\_pin = 24

s3\_pin = 25

s4\_pin = 26

# Initialize GPIO pins

GPIO.setup(pwm1\_pin, GPIO.OUT)

GPIO.setup(pwm2\_pin, GPIO.OUT)

GPIO.setup(s0\_pin, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(s1\_pin, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(s2\_pin, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(s3\_pin, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(s4\_pin, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

# Create PWM instances for servo control

pwm1 = GPIO.PWM(pwm1\_pin, 50)  # 50Hz frequency

pwm2 = GPIO.PWM(pwm2\_pin, 50)

# Define servo positions (adjust duty cycles as needed)

initial\_position = 7.5

servo1\_position = initial\_position

servo2\_position = initial\_position

# Start the PWM signals

pwm1.start(servo1\_position)

pwm2.start(servo2\_position)

def reset\_servos():

    global servo1\_position, servo2\_position

    servo1\_position = initial\_position

    servo2\_position = initial\_position

    pwm1.ChangeDutyCycle(servo1\_position)

    pwm2.ChangeDutyCycle(servo2\_position)

try:

    while True:

        if not GPIO.input(s0\_pin):

            servo2\_position = initial\_position + 2.5  # Clockwise 50%

            pwm2.ChangeDutyCycle(servo2\_position)

        if not GPIO.input(s4\_pin):

            servo2\_position = initial\_position - 2.5  # Anti-clockwise 50%

            pwm2.ChangeDutyCycle(servo2\_position)

        if not GPIO.input(s3\_pin):

            servo1\_position = initial\_position + 2.5  # Clockwise 50%

            pwm1.ChangeDutyCycle(servo1\_position)

        if not GPIO.input(s1\_pin):

            servo1\_position = initial\_position - 2.5  # Anti-clockwise 50%

            pwm1.ChangeDutyCycle(servo1\_position)

        if not GPIO.input(s2\_pin):

            reset\_servos()

except KeyboardInterrupt:

    pass

# Clean up GPIO on script exit

pwm1.stop()

pwm2.stop()

GPIO.cleanup()