**Πείραμα με διακόπτη 10 θέσεων**

1. **Περιγραφή:**

Οι περιστροφικοί διακόπτες μπορούν να εξηγηθούν σχετικά εύκολα. Πολλές συσκευές και κυκλώματα απαιτούν διακόπτες με πολλαπλές διαθέσιμες θέσεις, για την επιλογή διαφορετικών ηλεκτρικών κυκλωμάτων ή καταστάσεων εντός μιας συσκευής. Όπως υποδηλώνει και το όνομά τους, κινούνται μεταξύ αυτών των θέσεων μέσω περιστροφής.

Καθώς οι συμβατικοί διακόπτες μπορούν συνήθως να έχουν πρόσβαση μόνο σε δύο θέσεις (π.χ. on ή off), πολύπλοκες διαμορφώσεις ή συσκευές μπορεί να απαιτούν πολλαπλούς διακόπτες. Ένας περιστροφικός διακόπτης παρέχει μια απλούστερη, ολοκληρωμένη λύση.

Στο πείραμα μας χρησιμοποιώ ένα τέτοιο διακόπτη , σε συνδυασμό με κώδικα σε Python, ρυθμίζει τις ακίδες GPIO, διαβάζει τις θέσεις των διακοπτών και εκτυπώνει τη θέση στην κονσόλα. Μπορείτε να τροποποιήσετε τον κώδικα ώστε να εκτελείτε διαφορετικές ενέργειες με βάση τη θέση του διακόπτη, όπως τον έλεγχο των LED, των μοτέρ ή άλλων συσκευών.

1. **Υλικά:**

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

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **1\* Breadboard** | **Jumper Wires** | **1\* binary coded decimal, ten position rotary switch** |

1. **Γνώση υλικού:**

Ένας περιστροφικός κωδικοποιημένος διακόπτης χρησιμοποιείται για την επιλογή μιας από διάφορες επιλογές. Αυτή η συσκευή είναι σαν περιστροφικός διακόπτης, με τη διαφορά ότι οι έξοδοι είναι δυαδικά κωδικοποιημένες για να μειωθεί ο αριθμός των απαιτούμενων τερματικών. Οι περιστροφικοί κωδικοποιημένοι διακόπτες ενεργοποιούνται από έναν περιστρεφόμενο άξονα και έχουν διάφορες θέσεις. Τοποθετούνται σε πλακέτες τυπωμένων κυκλωμάτων (PCB) προκειμένου να ελέγχεται η έξοδός τους. Ορισμένοι περιστροφικοί κωδικοποιημένοι διακόπτες ενεργοποιούνται με κατσαβίδι, ενώ άλλοι περιλαμβάνουν τροχούς αντίχειρα ή τροχούς ώθησης που περιστρέφονται χειροκίνητα. Μπορούν να σχεδιαστούν είτε για να περιστρέφονται συνεχώς κατά 360 μοίρες, είτε για να σταματούν σε προκαθορισμένες θέσεις καθώς περιστρέφονται.

Οι περιστροφικοί κωδικοποιημένοι διακόπτες έχουν πολλά πλεονεκτήματα που περιλαμβάνουν τη μηχανική τους διάρκεια ζωής, τον αριθμό των θέσεων, μπορούν να περιστρέφονται κατά 360 μοίρες ή να σταματούν σε προκαθορισμένες θέσεις καθώς περιστρέφονται και μπορούν να χρησιμοποιηθούν σε εφαρμογές όπως συσκευές αμφίδρομης ραδιοεπικοινωνίας, συσκευές στόχευσης στόχων, αναμεταδότες αεροσκαφών, ιατρικός εξοπλισμός και εξοπλισμός βιομηχανικού αυτοματισμού.

|  |  |
| --- | --- |
| Διακόπτης 8 θέσεων | Διακόπτης 16 θέσεων |
|  |  |

**Πειραματική συσκευή:**

■ Πλακέτα ελέγχου Raspberry Pi: 1

■ Μονάδα διακόπτη binary coded decimal ten position rotary switch: 1

■ Γραμμές σύνδεσης

**Πειραματική σύνδεση:**

|  |  |
| --- | --- |
| Switch module | Raspberry Pi |
| VCC | 17 |
| GND | GND |
| switch module output (1,2,4,8) | 18 , 19 ,20 ,21 |

1. **Πειραματικό συμπέρασμα:**

Αυτός ο κώδικας διαβάζει τη δυαδική κωδικοποιημένη έξοδο από τον περιστροφικό διακόπτη και τη μετατρέπει σε δεκαδική τιμή. Στη συνέχεια, εκτυπώνει τη θέση στην κονσόλα κάθε φορά που ο διακόπτης αλλάζει θέση.

**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

# Set the GPIO mode

GPIO.setmode(GPIO.BCM)

# Define the GPIO pins for the switch and initialize them

switch\_common\_pin = 17

output\_pins = [18, 19, 20, 21]

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

for pin in output\_pins:

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

# Function to calculate the position

def calculate\_position():

    position = 0

    for i, pin in enumerate(output\_pins):

        if GPIO.input(pin) == GPIO.LOW:

            position += 2 \*\* i

    return position

try:

    while True:

        position = calculate\_position()

        print(f"Switch is in position {position}")

        GPIO.wait\_for\_edge(switch\_common\_pin, GPIO.BOTH)

except KeyboardInterrupt:

    GPIO.cleanup()  # Clean up GPIO on Ctrl+C

finally:

    GPIO.cleanup()  # Clean up GPIO on script exit