**Πείραμα με Self-Locking-Switch**

1. **Περιγραφή:**

Στο τρέχον πείραμα θα χρησιμοιήσουμε ένα διακόπτη self locking switch για να δω το πώς συμπεριφέρεται σε ένα ηλεκτρονικό κύκλωμα με τη βοήθεια του Raspberry Pi και του ανάλογου κώδικα σε Python.

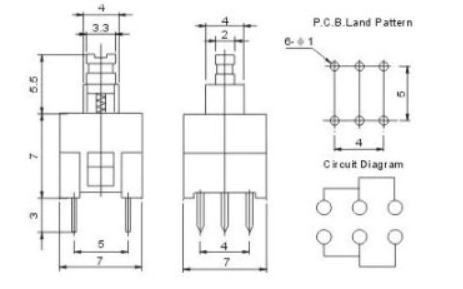
1. **Υλικά:**

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

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **1\* Breadboard** | **Jumper Wires** | **1\*self-locking switch module** |

1. **Γνώση υλικού:**

Διακόπτης με κουμπί αυτόματου κλειδώματος: Αυτός είναι ένας τύπος στιγμιαίου διακόπτη που παραμένει στην τελευταία του θέση αφού τον αφήσετε. Όταν πατάτε το κουμπί, αλλάζει κατάσταση (για παράδειγμα, από OFF σε ON) και παραμένει σε αυτή την κατάσταση μέχρι να το πατήσετε ξανά, με αποτέλεσμα να επανέλθει στην αρχική του κατάσταση (από ON σε OFF). Αυτοί οι διακόπτες χρησιμοποιούνται συχνά σε διάφορες ηλεκτρονικές εφαρμογές, όπως χειριστήρια διεπαφής χρήστη και διακόπτες ισχύος.



**Πειραματική σύνδεση:**

|  |  |
| --- | --- |
| The Self-Locking switch module | Raspberry Pi |
| Vcc | 5V |
| Gd | GND |
| PV1 , PD1 | 17 , 18 |

1. **Πειραματικό συμπέρασμα:**

Σε αυτό το παράδειγμα, ο κώδικας ελέγχει συνεχώς την κατάσταση του διακόπτη παρακολουθώντας τις ακίδες GPIO που είναι συνδεδεμένες στις εξόδους Pv1, Pd1 της μονάδας. Εκτυπώνει "Ο διακόπτης είναι πατημένος” όταν πατάμε το διακόπτη και το αντίστροφο.

**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 and pins

GPIO.setmode(GPIO.BCM)

pv\_pin = 17

pd\_pin = 18

# Setup GPIO pins

GPIO.setup(pv\_pin, GPIO.IN,GPIO.PUD\_DOWN)

GPIO.setup(pd\_pin, GPIO.IN,GPIO.PUD\_DOWN)

try:

    while True:

        # Check the state of the self-locking switch

        pv\_state = GPIO.input(pv\_pin)

        pd\_state = GPIO.input(pd\_pin)

        if pv\_state == GPIO.HIGH:

            print("Switch is not pressed (pv is HIGH)")

            # Your code for the action when the switch is pressed

        if pd\_state == GPIO.HIGH:

            print("Switch is  pressed (pd is HIGH)")

            # Your code for the action when the switch is not pressed

        time.sleep(0.1)  # Add a small delay to avoid rapid state changes

except KeyboardInterrupt:

    print("Program terminated by user")

finally:

    GPIO.cleanup()  # Clean up GPIO on exit

**Παραλλαγή πειράματος:**

Όταν πιέζουμε τον διακόπτη ενεργοποιείται βομβητής .

**Πειραματική σύνδεση:**

|  |  |
| --- | --- |
| The Self-Locking switch module | Raspberry Pi |
| Vcc | 5V |
| Gd | GND |
| PV1 , PD1 | 17 , 18 |

|  |  |
| --- | --- |
| The Buzzer module | Raspberry Pi |
| Vcc | 5V |
| Gd | GND |
| S | 22 |

import RPi.GPIO as GPIO

import time

# Set the GPIO mode and pins

GPIO.setmode(GPIO.BCM)

pv\_pin = 17

pd\_pin = 18

buzzer\_pin = 22  # Change this to the GPIO pin connected to the buzzer signal (S)

# Setup GPIO pins

GPIO.setup(pv\_pin, GPIO.IN,GPIO.PUD\_DOWN)

GPIO.setup(pd\_pin, GPIO.IN,GPIO.PUD\_DOWN)

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

def activate\_buzzer():

    GPIO.output(buzzer\_pin, GPIO.HIGH)

    time.sleep(0.5)  # Buzzer on for 0.5 seconds

    GPIO.output(buzzer\_pin, GPIO.LOW)

try:

    while True:

        # Check the state of the self-locking switch

        pv\_state = GPIO.input(pv\_pin)

        pd\_state = GPIO.input(pd\_pin)

        if pv\_state == GPIO.HIGH:

            print("Switch is not pressed (pv is HIGH)")

            # Your code for the action when the switch is not pressed

        if pd\_state == GPIO.HIGH:

            print("Switch is  pressed (pd is HIGH)")

            # Your code for the action when the switch is  pressed

            activate\_buzzer()

        time.sleep(0.1)  # Add a small delay to avoid rapid state changes

except KeyboardInterrupt:

    print("Program terminated by user")

finally:

    GPIO.cleanup()  # Clean up GPIO on exit