Κίνηση - Έλεγχος Κινητήρων

ΜΑΘΗΜΑΤΑ RASPBERRY

ΜΑΝΟΣ ΝΙΚΟΛΑΟΣ

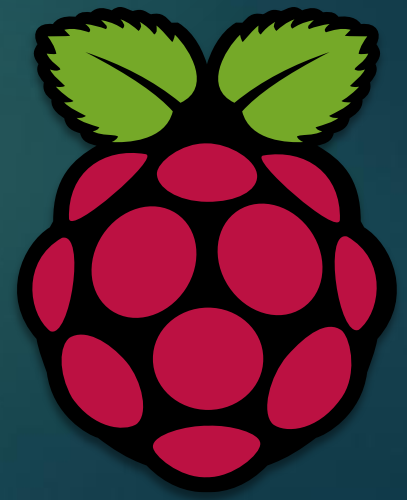
ΔΟΥΜΑ ΑΝΑΣΤΑΣΙΑ

ΚΑΒΑΛΛΙΕΡΑΤΟΥ ΕΡΓΙΝΑ

Περιεχόμενα

Εισαγωγή στους Κινητήρες

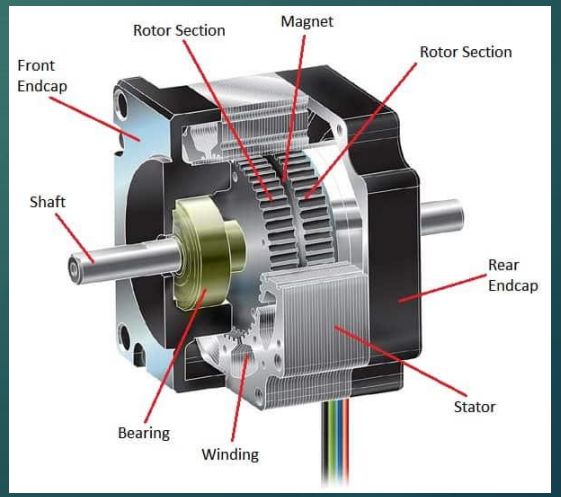
- Η χρήση των κινητήρων είναι απαραίτητη σε εφαρμογές ρομποτικών συστημάτων ή συστημάτων αυτοματισμού που απαιτούν κίνηση
- Κάθε κινητήρας εκτελεί κάποιες κινήσεις με σκοπό να κινηθεί ή να περιστραφεί το ρομπότ στο περιβάλλον το οποίο βρίσκεται
- Οι εντολές κίνησης και περιστροφής γίνονται μέσω της πλακέτας στην οποία είναι συνδεδεμένος ο κάθε κινητήρας



Κατηγορίες Κινητήρων

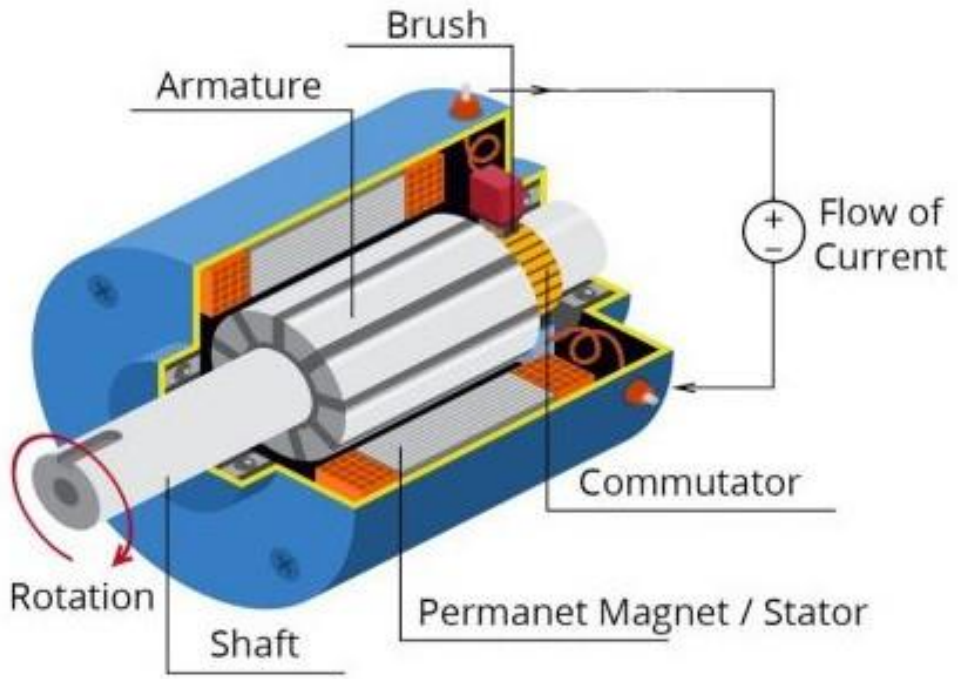
Οι κινητήρες μπορούν να ταξινομηθούν:

- Σε DC (συνεχούς ρεύματος), δηλαδή να κινούνται συνεχόμενα προς μια κατεύθυνση ή αντίθετα. Η κατηγορία αυτή είναι και η πιο κοινή
- Σε σερβοκινητήρες, οι οποίοι έχουν ένα εύρος μοιρών που μπορούν να περιστραφούν, οι πιο κοινές τιμές μοιρών είναι 90°, 180°, 360°
- Τέλος, σε βηματικούς, δηλαδή σε αυτούς που μετατρέπουν μια παλμοσειρά που στρέφει τον άξονα κατά μια σταθερή γωνία



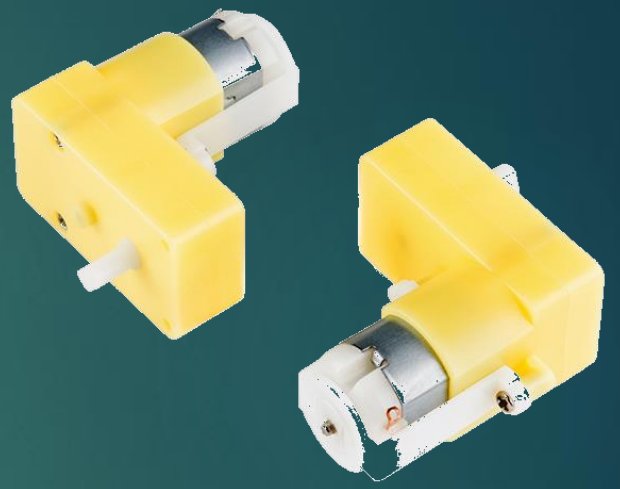
DC κινητήρες

- Οι DC κινητήρες περιλαμβάνονται από τον ρότορα ή δρομέα που αποτελεί το κινούμενο μέρος, τον στάτη που αποτελείται από μόνιμους ή τεχνητούς μαγνήτες και τις ψήκτρες (ή καρβουνάκια)
- Υπάρχουν διάφορες διατάξεις και μεγέθη DC κινητήρων ανάλογα με την περίπτωση ή την κατασκευή που θα χρειαστεί να υλοποιηθεί



Hobby Motor (DC)

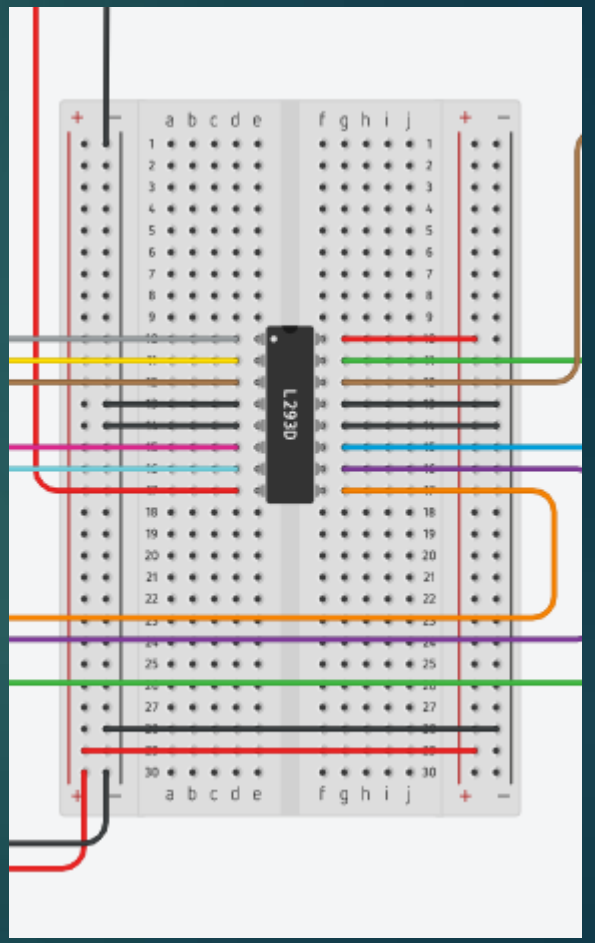
- Χαρακτηριστικό παράδειγμα DC κινητήρα αποτελούν οι hobby motors οι οποίοι χρησιμοποιούνται σε κατασκευές για την κίνηση μικρών ρομποτικών αμαξιδίων και έχουν διάφορες διατάξεις
- Διαθέτουν εσωτερικά γρανάζια έτσι ώστε να έχουν την κατάλληλη ροπή για να κινήσουν μικρές κατασκευές
- Η τάση εισόδου κυμαίνεται από 3 έως 6 volt, ενώ καταναλώνουν 100mA και η ταχύτητα τους ανέρχεται στα 125RPM



Συνδεσμολογία DC κινητήρα και L293D

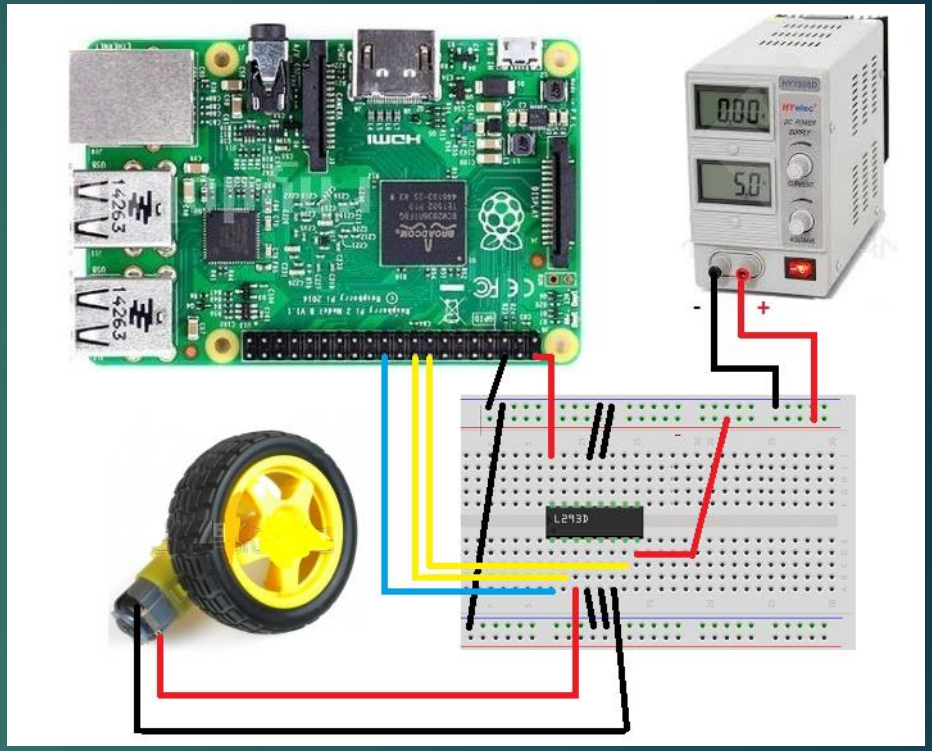
- Όταν απαιτούνται μεγαλύτεροι κινητήρες και δεν επαρκεί η τάση της πλακέτας χρησιμοποιείται ένας επιπλέον driver (H-bridge), όπως το ολοκληρωμένο L293D, ώστε να επιτευχθεί η εξωτερική τροφοδοσία από μπαταρίες ή τροφοδοτικό

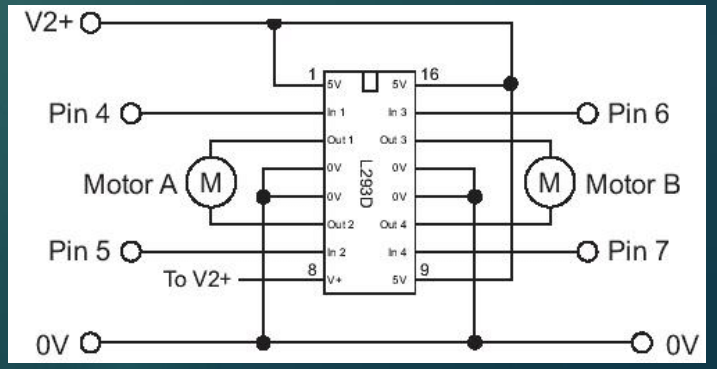
- Δεξιά παρουσιάζεται η συνδεσμολογία του ολοκληρωμένου L293D, κατά την οποία συνδέονται οι γειώσεις, η τάση της εξωτερικής πηγής, η τάση του ολοκληρωμένου, οι κινητήρες και τα σήματα εισόδου από την πλακέτα. Ακολουθεί αναλυτικό παράδειγμα στις επόμενες διαφάνειες



Παράδειγμα με L293D

- Οι γειώσεις συνδέονται στα pin 4, 5, 12 και 13
- Η εξωτερική τάση συνδέεται στο pin 8, ενώ τάση του ολοκληρωμένου συνδέεται στο pin 16
- Τα pin 1 και 9 χρησιμοποιούνται για την ενεργοποίηση των δύο καναλιών για τους κινητήρες, επομένως θα συνδεθούν στα 5v ή σε 2 GPIO pin που θα δίνουν HIGH (5v)
- Τα pin 2 και 7 θα συνδεθούν σε δύο GPIO θύρες του raspberry για να δώσουν σήμα στον έναν κινητήρα και τα 10 και 15 θα συνδεθούν σε άλλα 2 GPIO για τον δεύτερο κινητήρα
- Τέλος, ο ένας κινητήρας θα συνδεθεί στα pin 3 και 6, ενώ ο δεύτερος στα 11 και 14



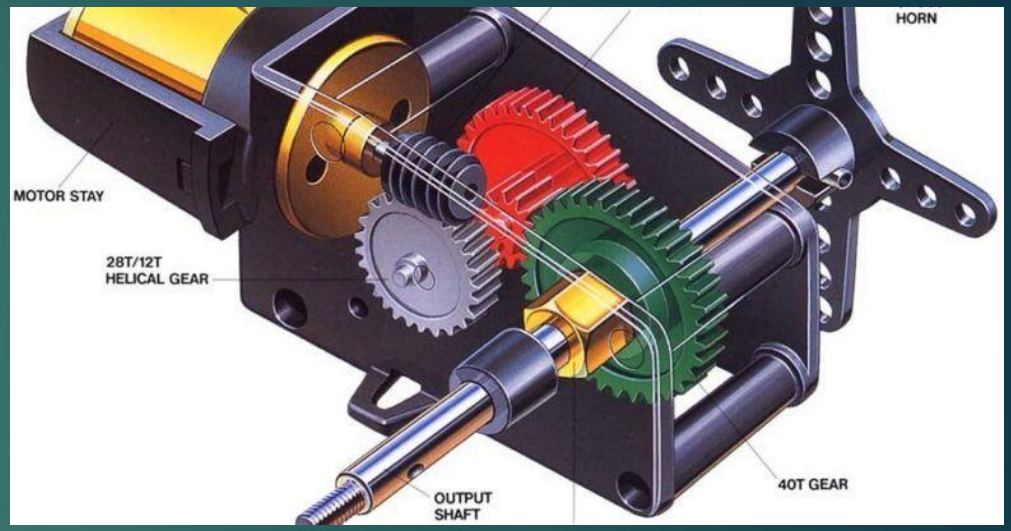


Κώδικας


```
#εισαγωγή και ρύθμιση GPIO θυρών
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)

#ορισμός pin κινητήρα & enable pin
Motor1 = 16    # Input Pin
Motor2 = 18    # Input Pin
Motor3 = 22    # Enable Pin

#ορισμός GPIO ως έξοδος
GPIO.setup(Motor1,GPIO.OUT)
GPIO.setup(Motor2,GPIO.OUT)
GPIO.setup(Motor3,GPIO.OUT)

#κίνηση μπροστά
print "FORWARD MOTION"
GPIO.output(Motor1,GPIO.HIGH)
GPIO.output(Motor2,GPIO.LOW)
GPIO.output(Motor3,GPIO.HIGH)

sleep(3)

#κίνηση πίσω
print "BACKWARD MOTION"
GPIO.output(Motor1,GPIO.LOW)
GPIO.output(Motor2,GPIO.HIGH)
GPIO.output(Motor3,GPIO.HIGH)

sleep(3)
#καμία κίνηση
print "STOP"
GPIO.output(Motor3,GPIO.LOW)

GPIO.cleanup()
```

Σερβοκινητήρας

- Η λειτουργία του σερβοκινητήρα ή αλλιώς servo motor συνδιάζεται με την ανάδραση της θέσης και κάποιες φορές και την ταχύτητα του
- Αποτελείται από έναν κατάλληλο κινητήρα συζευγμένο με έναν αισθητήρα για την ανατροφοδότηση θέσεως. Για το λόγο αυτό υπάρχουν ένα ή δύο επιπλέον καλώδια ελέγχου σήματος



Μοντέλα σερβοκινητήρων

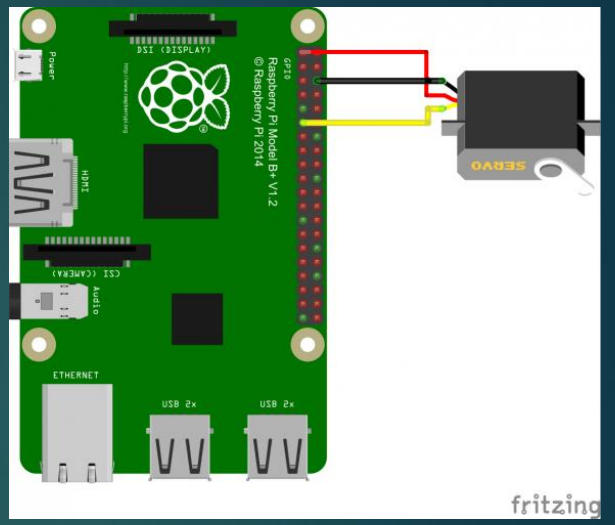
- Δύο από τα πιο κοινά μοντέλα σερβοκινητήρων είναι το tower pro 9g και το Hitec HS-645MG που παρουσιάζονται στις εικόνες
- Το πρώτο μοντέλο έχει τάση λειτουργίας 4.8 - 6 volt και μπορεί να περιστραφεί κατά 180°
- Το δεύτερο μοντέλο έχει τάση λειτουργίας 4.8 - 6 volt, μπορεί να περιστραφεί κατά 197° αλλά ενδείκνυται για κατασκευές που απαιτούν μεγαλύτερη ροπή

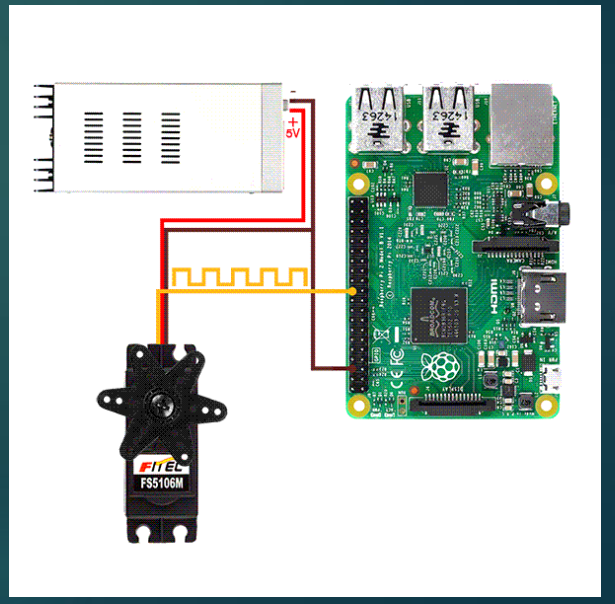
Συνδεσμολογία Σερβοκινητήρα

Η συνδεσμολογία ενός σερβοκινητήρα παρουσιάζεται παρακάτω:

- Το pin 1 συνδέεται στην γείωση της πλακέτας
- Το pin 2 συνδέεται στην τάση 5V
- Το pin 3 συνδέεται σε μία GPIO θύρα του Raspberry

Σε περίπτωση που χρειαστεί εξωτερική τροφοδοσία ο σερβοκινητήρας τότε η εξωτερική τάση θα συνδεθεί στο κόκκινο καλώδιο ενώ η γείωση θα είναι κοινή με το τροφοδοτικό και το raspberry, όπως φαίνεται στην κάτω δεξιά εικόνα.



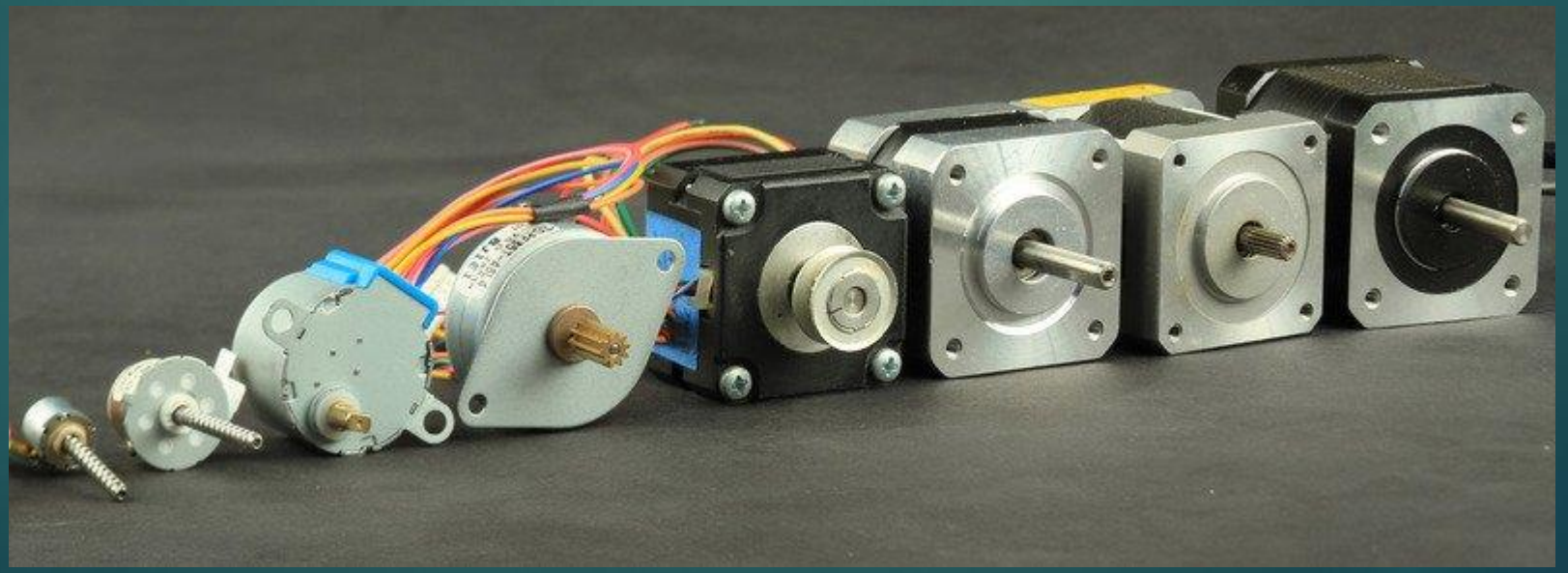


Κώδικας


```
#εισαγωγή και ρύθμιση GPIO θυρών
import RPi.GPIO as GPIO
import time

#δήλωση pin servo
servoPIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(servoPIN, GPIO.OUT)

# GPIO για σήμα PWM με συχνότητα 50Hz
p = GPIO.PWM(servoPIN, 50)
p.start(2.5) # αρχικοποίηση
try:
  while True:
    #αλλαγή θέσης κάθε 0.5 sec
    p.ChangeDutyCycle(5)
    time.sleep(0.5)
    p.ChangeDutyCycle(7.5)
    time.sleep(0.5)
    p.ChangeDutyCycle(10)
    time.sleep(0.5)
    p.ChangeDutyCycle(12.5)
    time.sleep(0.5)
    p.ChangeDutyCycle(10)
    time.sleep(0.5)
    p.ChangeDutyCycle(7.5)
    time.sleep(0.5)
    p.ChangeDutyCycle(5)
    time.sleep(0.5)
    p.ChangeDutyCycle(2.5)
    time.sleep(0.5)
except KeyboardInterrupt:
  p.stop()
  GPIO.cleanup()
```

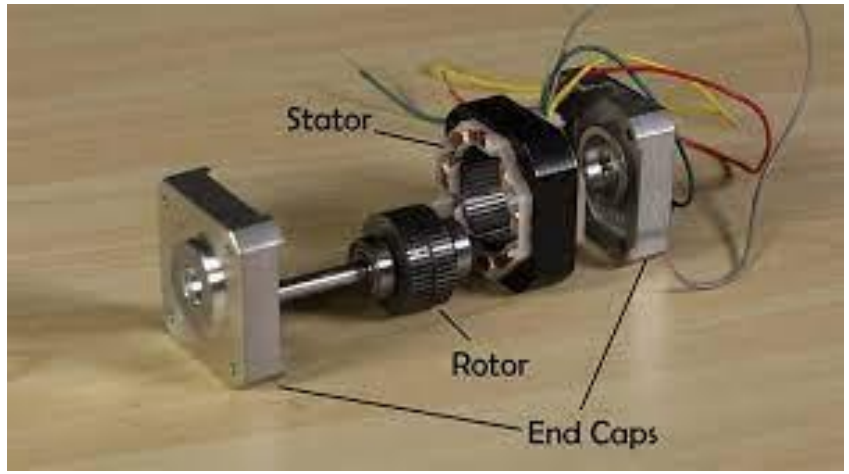
Βηματικός Κινητήρας (1/2)

- Ο βηματικός κινητήρας ή αλλιώς stepper motor είναι ένας κινητήρας επαγωγής, που έχει ακρίβεια ανά βήμα
- Κάθε φορά που έρχεται ένας παλμός ο άξονας περιστρέφεται κατά μία ιδιαίτερη γωνία (γωνία βημάτων)
- Αφού ο άξονας περιστραφεί κατά ένα βήμα, σταματά μέχρι να πάρει την επόμενη εντολή οδήγησης



Βηματικός Κινητήρας (2/2)

- Η αρχιτεκτονική του βηματικού κινητήρα φαίνεται στη δεξιά εικόνα
- Διαθέτει 4, 5 ή 6 καλώδια εκ των οποίων:
  - Το ένα ζεύγος συνδέεται με το ένα πηνίο εσωτερικά του κινητήρα
  - Το δεύτερο ζεύγος με το δεύτερο πηνίο του κινητήρα
  - Τέλος, σε κάποιους κινητήρες υπάρχει ένα ή δύο καλώδια σε κάποιο κοινό σημείο (com) και συνδέονται στην τάση ή τη γείωση ανάλογα με το μοντέλο του κινητήρα

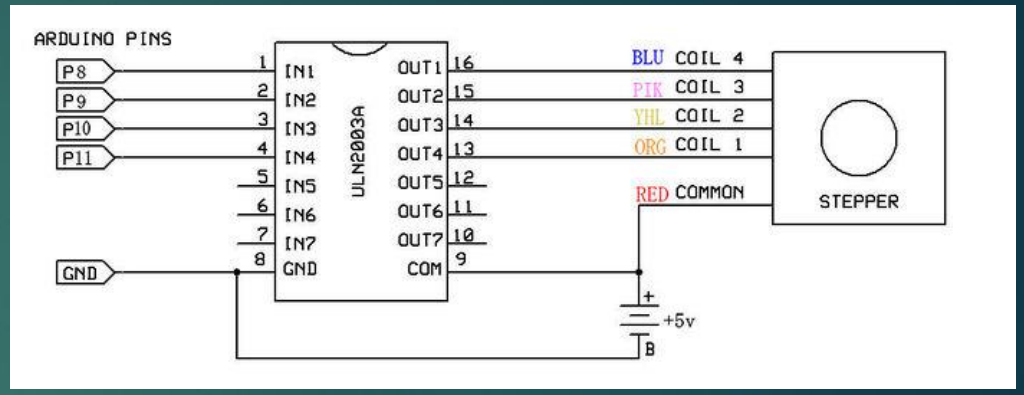


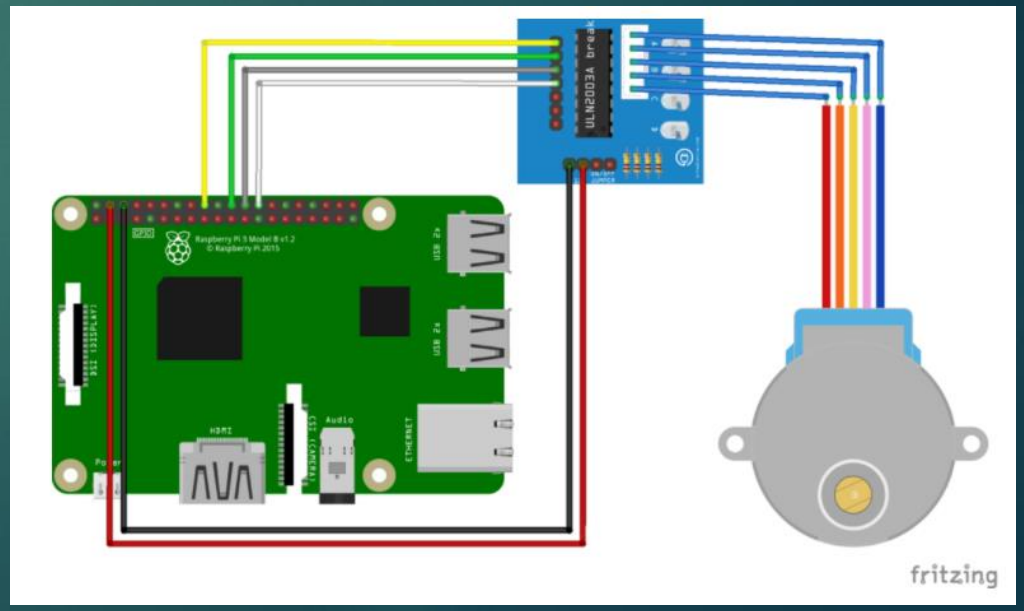
Stepper Motor 42BYGHW804

- Χαρακτηριστικό μοντέλο βηματικού κινητήρα είναι το 42BYGHW804 ή NEMA 17 το οποίο χρησιμοποιείται σε 3d-printers αλλά και σε διάφορες κατασκευές - συσκευές που απαιτούν ακρίβεια στην κίνηση
- Έχει τα εξής χαρακτηριστηκά:
  - Τάση: 3.6 volt
  - Ένταση ρεύματος: 1.2 A/φάση
  - Γωνία βήματος: 1.8°
  - Ροπή συγκράτησης: 4.8kg.cm
  - Διάμετρος άξονα: 5mm

Βηματικός κινητήρας και ULN2003

- Για τη συνδεσμολογία ενός βηματικού κινητήρα με εξωτερική τροφοδοσία θα χρειαστεί το ολοκληρωμένο driver ULN2003 στο οποίο ακολουθείται η παρακάτω συνδεσμολογία:
  - Τα 4 καλώδια του βηματικού κινητήρα θα συνδεθούν στα 4 out του ULN2003, ενώ το κοινό καλώδιο στο COM
  - Θα χρησιμοποιηθούν 4 από τις GPIO θύρες του Raspberry ώστε να συνδεθούν στα 4 input του ολοκληρωμένου
  - Η εξωτερική τάση θα συνδεθεί στο COM ή εναλλακτικά με τα +5v του raspberry
  - Το GND θα συνδεθεί σε κοινή γείωση του raspberry και της εξωτερικής πηγής





Κώδικας (1/3)


```
# εισαγωγή βιβλιοθηκών
import time
import RPi.GPIO as GPIO

# ορισμός των pin ως GPIO
GPIO.setmode(GPIO.BCM)

# αρχικοποίηση GPIO σημάτων
StepPins = [24,25,8,7]
# ρύθμιση όλων των Pin ως output
for pin in StepPins:
        print("Setup pins")
        GPIO.setup(pin,GPIO.OUT)
        GPIO.output(pin, False)
# ορισμός χρόνου αναμονής
WaitTime = 0.005
```

```
# ορισμός ακολουθιών
StepCount1 = 4
Seq1 = []
Seq1 = [i for i in range(0, StepCount1)]
Seq1[0] = [1,0,0,0]
Seq1[1] = [0,1,0,0]
Seq1[2] = [0,0,1,0]
Seq1[3] = [0,0,0,1]
# ορισμός σύνθετων ακολουθιών
StepCount2 = 8
Seq2 = []
Seq2 = [i for i in range(0, StepCount2)]
Seq2[0] = [1,0,0,0]
Seq2[1] = [1,1,0,0]
Seq2[2] = [0,1,0,0]
Seq2[3] = [0,1,1,0]
Seq2[4] = [0,0,1,0]
Seq2[5] = [0,0,1,1]
Seq2[6] = [0,0,0,1]
Seq2[7] = [1,0,0,1]
```

Κώδικας (2/3)


```
# επιλογή ακολουθίας
Seq = Seq2
StepCount = StepCount2

def steps(nb):
        StepCounter = 0
        if nb<0: sign=-1
        else: sign=1
        nb=sign*nb*2 #επί 2 για το μισό βήμα
        print("nbsteps {} and sign {}".format(nb,sign))
        for i in range(nb):
                for pin in range(4):
                        xpin = StepPins[pin]
                        if Seq[StepCounter][pin]!=0:
                                GPIO.output(xpin, True)
                        else:
                                GPIO.output(xpin, False)
                StepCounter += sign
```

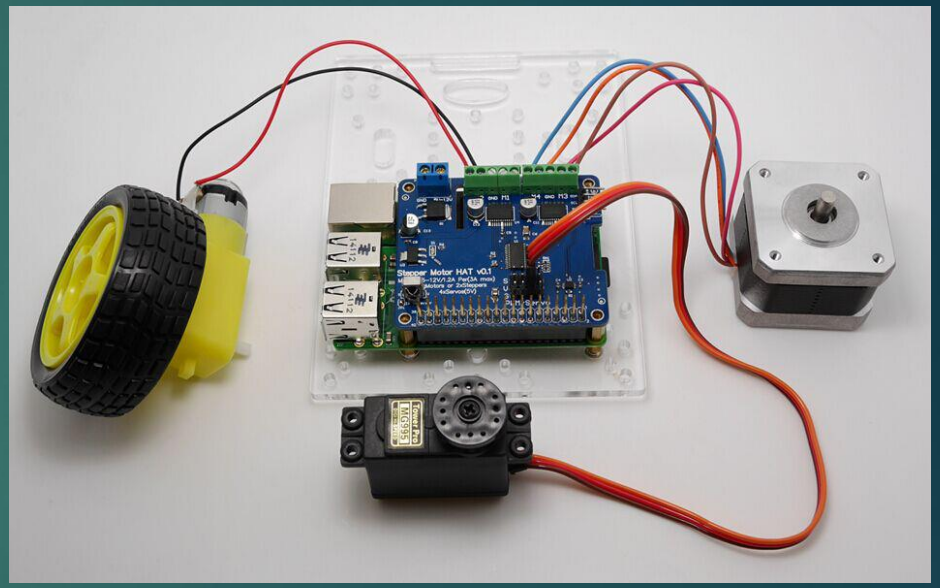
```
# Αν φτάσουμε στο τέλος της ακολουθίας
        # εκκίνηση ξανά
                if (StepCounter==StepCount):
                        StepCounter = 0
                if (StepCounter<0):
                        StepCounter = StepCount-1
                # καθυστέρηση
                time.sleep(WaitTime)
```

Κώδικας (3/3)


```
# συνάρτηση main
nbStepsPerRev=2048
if __name__ == '__main__' :
    hasRun=False
    while not hasRun:
        steps(nbStepsPerRev)
        time.sleep(1)
        steps(-nbStepsPerRev)
        time.sleep(1)
        hasRun=True
    print("Stop motor")
    for pin in StepPins:
        GPIO.output(pin, False)
```

Συνδυαστικό παράδειγμα

- Στο παράδειγμα που ακολουθεί συνδυάζονται και τα 3 είδη κινητήρων με τη βοήθεια μιας πλακέτας επέκτασης (MotorHat)
- Στο hat συνδέονται ένας DC hobby motor, ένας servo motor HS-645MG και ένας stepper motor Nema 17
- Η τάση λειτουργίας είναι στα 5 - 12 volt



Συνδυαστικό παράδειγμα - κώδικας


```
from MotorHAT import MotorHAT, DCMotor
import time
import atexit
# δημιουργία default αντικειμένου
mh = MotorHAT(addr=0x6f)  # συνιστάται για αυτόματη απενεργοποίηση
def turnOffMotors():
mh.getMotor(1).run(MotorHAT.RELEASE)
mh.getMotor(2).run(MotorHAT.RELEASE)
mh.getMotor(3).run(MotorHAT.RELEASE)
mh.getMotor(4).run(MotorHAT.RELEASE)
atexit.register(turnOffMotors)
myMotor = mh.getMotor(1) # ρύθμιση ταχύτητας, από 0 (off) έως 255 (μέγιστη ταχύτητα)
myMotor.setSpeed(150)
myMotor.run(MotorHAT.FORWARD);
motormyMotor.run(MotorHAT.RELEASE); # ενεργοποίηση κινητήρα
```

Συνδυαστικό παράδειγμα - κώδικας


```
while (True):
print("Forward! ")
myMotor.run(MotorHAT.FORWARD)
print("\tSpeed up...")
for i in range(255):
        myMotor.setSpeed(i)
        time.sleep(0.01)
print("\tSlow down...")
for i in reversed(range(255)):
        myMotor.setSpeed(i)
        time.sleep(0.01)
print("Backward! ")
myMotor.run(MotorHAT.BACKWARD)
print("\tSpeed up...")
for i in range(255):
        myMotor.setSpeed(i)
        time.sleep(0.01)
print("\tSlow down...")
for i in reversed(range(255)):
        myMotor.setSpeed(i)
        time.sleep(0.01)
print("Release")
myMotor.run(MotorHAT.RELEASE)
time.sleep(1.0)
```

Συνδυαστικό παράδειγμα - κώδικας


```
from __future__ import division
import time
import atexit
# εισαγωγή του PCA9685 module.
from MotorHAT.PWM_Servo_Driver import PWM
# αρχικοποίηση του PCA9685 χρησιμοποιώντας την default διεύθυνση (0x40).
pwm = PWM(address=0x6f)
def turnOffServos():
        pwm.setPWM(0,0,4096)
        pwm.setPWM(1,0,4096)
        pwm.setPWM(14,0,4096)
        pwm.setPWM(15,0,4096)
atexit.register(turnOffServos)
servo_min = 150  # ελάχιστο μήκος παλμού από 4096
servo_max = 600  # μέγιστο μήκος παλμού από 4096
```

Συνδυαστικό παράδειγμα - κώδικας


```
# βοηθητική συνάρτηση για ευκολότερη ρύθμιση πλάτους παλμού servo
def set_servo_pulse(channel, pulse):
    pulse_length = 1000000
    pulse_length //= 60       # 60 Hz
    print('{0}us per period'.format(pulse_length))
    pulse_length //= 4096
    print('{0}us per bit'.format(pulse_length))
    pulse *= 1000
    pulse //= pulse_length
    pwm.setPWM(channel, 0, pulse)
```

```
# ρύθμιση συχνότητας στα 60hz
pwm.setPWMFreq(60)

print('Moving servo on channel 0, press Ctrl-C to quit...')
while True:
    # μετακίνηση servo στο κανάλι 0
    pwm.setPWM(0, 0, servo_min)
    time.sleep(1)
    pwm.setPWM(0, 0, servo_max)
    time.sleep(1)
```

Συνδυαστικό παράδειγμα - κώδικας


```
from MotorHAT import MotorHAT, DCMotor, StepperMotor
import time
import atexit
# δημιουργία default αντικειμένου, χωρίς αλλαγές στην I2C διεύθυνση ή συχνότητα
mh = MotorHAT(addr=0x6f)# προτείνεται για αυτόματη απενεργοποίηση κινητήρων
def turnOffMotors():
        mh.getMotor(1).run(MotorHAT.RELEASE)
        mh.getMotor(2).run(MotorHAT.RELEASE)
        mh.getMotor(3).run(MotorHAT.RELEASE)
        mh.getMotor(4).run(MotorHAT.RELEASE)
atexit.register(turnOffMotors)myStepper = mh.getStepper(200, 1)
# 200 steps/rev, θύρα κινητήρα #1
myStepper.setSpeed(50)
```

Συνδυαστικό παράδειγμα - κώδικας


```
while (True):
    print("Single coil steps")
    myStepper.step(100, MotorHAT.FORWARD,  MotorHAT.SINGLE)
    myStepper.step(100, MotorHAT.BACKWARD, MotorHAT.SINGLE)

    print("Double coil steps")
    myStepper.step(100, MotorHAT.FORWARD,  MotorHAT.DOUBLE)
    myStepper.step(100, MotorHAT.BACKWARD, MotorHAT.DOUBLE)

    print("Interleaved coil steps")
    myStepper.step(100, MotorHAT.FORWARD,  MotorHAT.INTERLEAVE)
    myStepper.step(100, MotorHAT.BACKWARD, MotorHAT.INTERLEAVE)

    print("Microsteps")
    myStepper.step(100, MotorHAT.FORWARD,  MotorHAT.MICROSTEP)
    myStepper.step(100, MotorHAT.BACKWARD, MotorHAT.MICROSTEP)
```

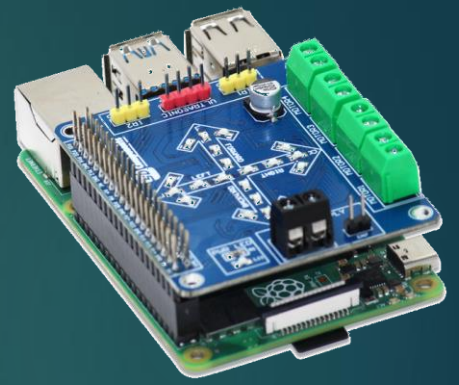
Σύγκριση Κινητήρων

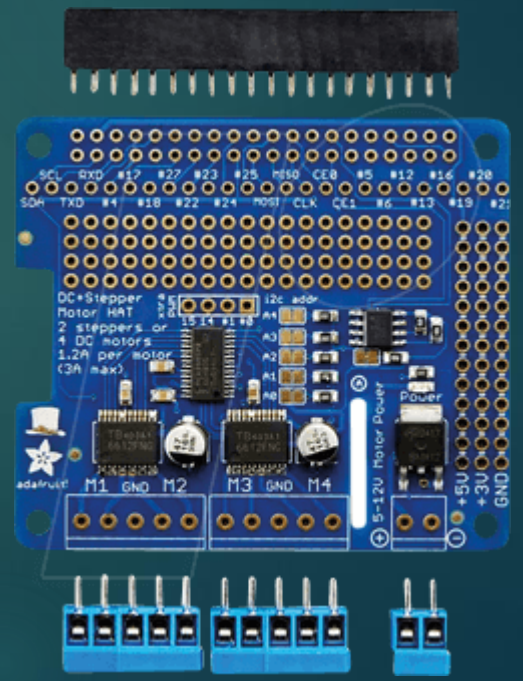
- Servo vs stepper
  - Οι σερβοκινητήρες καταναλώνουν ενέργεια μόνο όσο περιστρέφονται, ενώ οι βηματικοί συνεχίζουν να καταναλώνουν ενέργεια για να διατηρήσουν τη θέση τους
  - Ο ελεγκτής πρέπει να «γνωρίζει» τη θέση του βηματικού κινητήρα κατά την ενεργοποίηση. Ένας σερβοκινητήρας αντιθέτως κινείται αμέσως σε οποιαδήποτε γωνία, ανεξάρτητα από την αρχική του θέση κατά την ενεργοποίηση
  - Η έλλειψη ανάδρασης του βηματικού κινητήρα μπορεί να οδηγήσει σε σφάλματα θέσης και το σύστημα μπορεί να χρειαστεί να επαναρυθμιστεί ή να γίνει επανεκκίνηση. Ο κωδικοποιητής και ο ελεγκτής ενός σερβοκινητήρα είναι ένα πρόσθετο κόστος, αλλά βελτιστοποιούν την απόδοση του συνολικού συστήματος

Επεκτάσεις

Hat Raspberry

- Υπάρχουν στο εμπόριο πλακέτες επέκτασης (Hat) για τη σύνδεση των κινητήρων με το Raspberry
- Σε αυτή την περίπτωση δεν χρειάζεται η συνδεσμολογία και η καλωδίωση με breadboard σύμφωνα με τις παραπάνω διαφάνειες
- Επομένως θα χρειαστεί μόνο η σύνδεση των κινητήρων (servo, stepper, DC) και το προγραμματιστικό κομμάτι



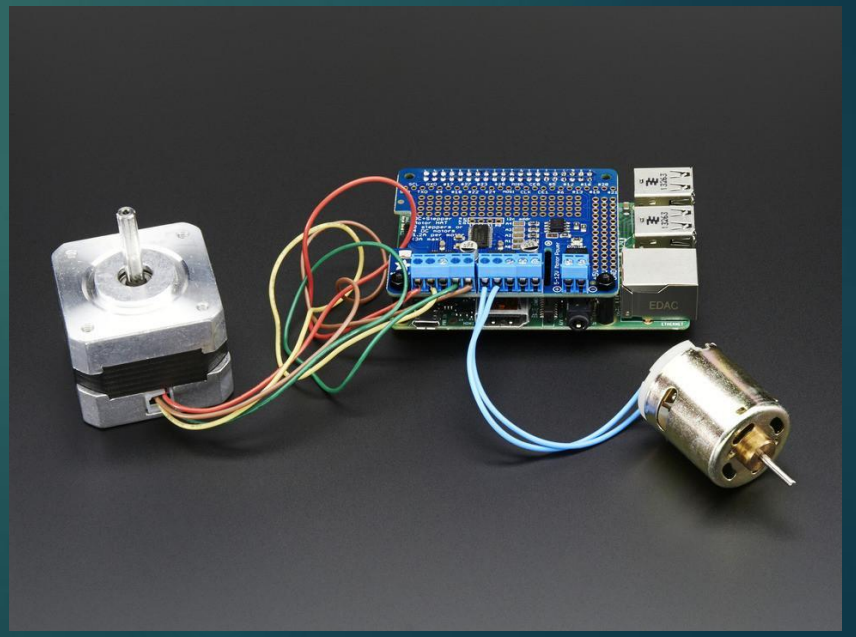


Προσοχή σε:

- **Τροφοδοσία**
  - Οι περισσότεροι κινητήρες λειτουργούν στα 5-6V ή 12V, για το λόγο αυτό χρειάζεται η σωστή συνδεσμολογία με την τάση. Σε περίπτωση που η τάση λειτουργίας κάποιου κινητήρα είναι διαφορετική των παραπάνω, τότε μπορούν να χρησιμοποιηθεί εξωτερική τροφοδοσία
- **Βαθμονόμηση (Calibration)**
  - Πολύ σημαντικό ρόλο παίζει η σωστή ρύθμιση των κινητήρων σε σχέση με την αρχική και την τελική θέση αν πρόκειται για συγκεκριμένη περιστοφή π.χ. για ρομποτικό βραχίονα
  - Απαιτούνται επομένως αρκετές δοκιμές έως ότου βγει το επιθυμητό αποτέλεσμα

Δυνατότητες

- Συμπερασματικά οι δυνατότητες που παρέχει το Raspberry σε συνδυασμό με το πλήθος των κινητήρων που υπάρχουν στην αγορά, μπορούν να υλοποιηθούν πρωτότυπες και πολύ χρήσιμες κατασκευές - εφαρμογές στα πλαίσια της ρομποτικής αλλά και του αυτοματισμού
- Περισσότερα συνδυαστικά παραδείγματα θα ακολουθήσουν στα δύο τελευταία μαθήματα



Βιβλιογραφία

https:/tutorials-raspberrypi.com/raspberry-pi-servo-motor-control/

https:/en.wikipedia.org/wiki/DC_motor

https:/www.rhydolabz.com/wiki/?p=11288

https:/www.digikey.com/en/maker/blogs/2021/how-to-control-servo-motors-with-a-raspberry-pi

https:/learn.adafruit.com/adafruits-raspberry-pi-lesson-8-using-a-servo-motor

https:/tutorials-raspberrypi.com/how-to-control-a-stepper-motor-with-raspberry-pi-and-l293d-uln2003a/

https:/www.aranacorp.com/en/control-a-stepper-with-raspberrypi/

https:/osoyoo.com/fr/2017/09/04/motor-hat-for-raspberry-pi/