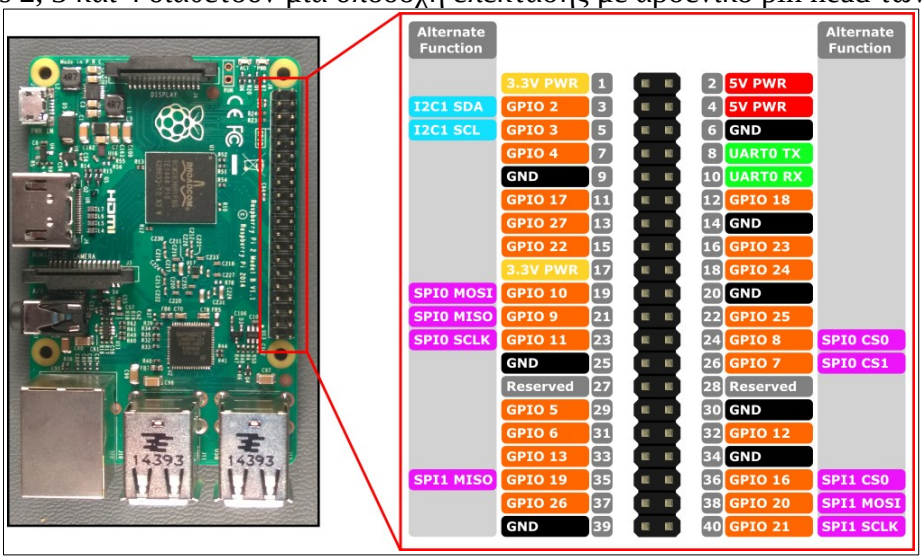
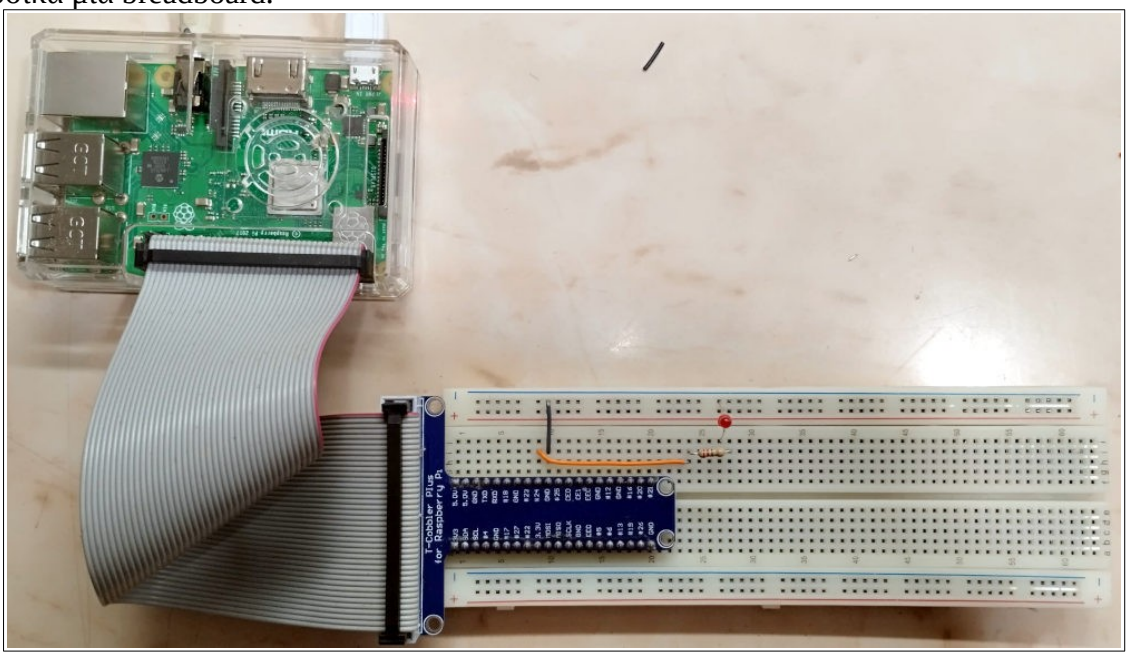
Προγραμματισμός υλικού σε Python

Η υποδοχή επέκτασης

Τα Raspberries 2, 3 και 4 διαθέτουν μια υποδοχή επέκτασης με αρσενικό pin head των 40 ακίδων.

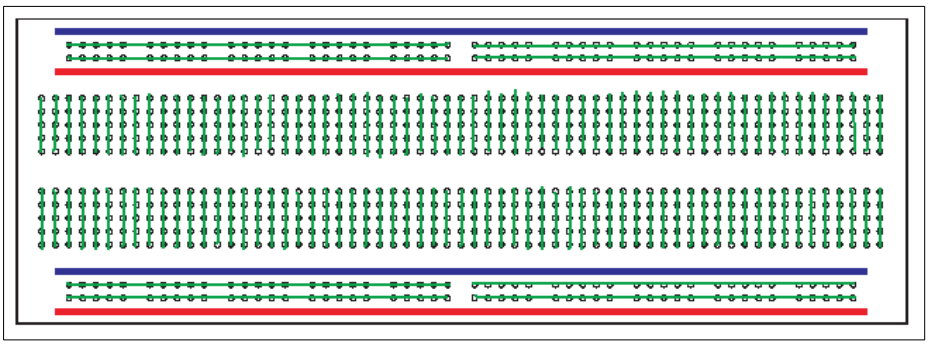


Για τα πειράματα μας θα χρησιμοποιήσουμε ένα μετατροπέα από αυτό το pin head σε breadboard και φυσικά μια breadboard.

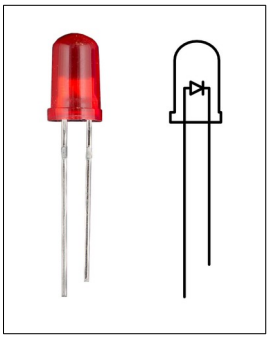


Η breadboard

Η breadboard είναι μια επιφάνεια με οπές στην οποία μπορούμε να φτιάχνουμε εύκολα κυκλώματα δίχως να χρειάζεται να κάνουμε κολλήσεις. Αρκεί μόνο να τοποθετούμε τους ακροδέκτες των εξαρτημάτων μέσα στις οπές. Γενικά οι breadboards ακολουθούν την διάταξη της παρακάτω εικόνας. Δηλαδή οι οπές είναι εσωτερικά συνδεδεμένες σύμφωνα με την εικόνα που ακολουθεί. Οι γραμμές μπλε – κόκκινο στην πάνω και κάτω πλευρά είναι οριζόντια ενωμένες και χρησιμοποιούνται συνήθως για τις τάσεις τροφοδοσίας ενώ οι υπόλοιπες είναι κάθετα ενωμένες σε πεντάδες. Η εσοχή στον κεντρικό άξονα διακόπτει τις κάθετες συνδέσεις.



1. Έξοδος με LED

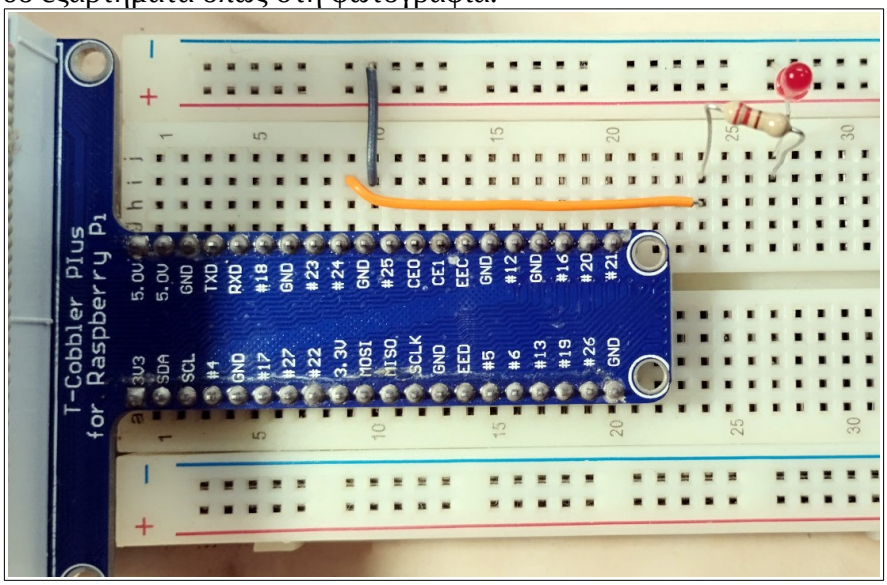


Το πιο απλό εξάρτημα για την εμφάνιση καταστάσεων μιας μηχανής είναι η ενδεικτική λυχνία. Σήμερα χρησιμοποιούμε τις λυχνίες L.E.D. γιατί παρουσιάζουν σημαντικά πλεονεκτήματα. Τα απλά Led είναι μονόχρωμα και έχουν δύο ακροδέκτες. Προσοχή, επειδή είναι δίοδοι πρέπει να πολωθούν σωστά ώστε να μπορούν να ανάψουν. Ο μεγαλύτερος ακροδέκτης είναι ο θετικός και ο μικρότερος είναι ο αρνητικός. Το μέγιστο ρεύμα που μπορεί να διαπεράσει τα ενδεικτικά LED είναι 20mA, το οποίο αν ξεπεραστεί τότε αυτό καταστρέφεται. Για να διασφαλίσουμε ότι δεν θα περάσει η ένταση του ρεύματος αυτή την την τιμή, πάντα βάζουμε σε σειρά με το LED έναν αντιστάτη από 100 – 330Ω. Μια συνηθισμένη τιμή είναι τα 220Ω σε τάση 3,3V. Όταν η ένταση είναι μεγαλύτερη τότε το LED φωτοβολεί πιο δυνατά.

1.1 Το πρώτο μας πείραμα

Το κύκλωμα

Συνδέουμε τα δύο εξαρτήματα όπως στη φωτογραφία.



Συγκεκριμένα με το γκρι σύρμα ενώσαμε την γείωση GND (0 Volts) του raspberry στην πάνω μπλε γραμμή που πάει οριζόντια και έχει ενωμένες όλες τις οπές οριζόντια. Με το πορτοκαλί ενώσαμε το σήμα **GPIO24** με την οπή h24. Οι οπές f24, g24, h24, i24, j24 είναι όλες ενωμένες μεταξύ τους. Μετά βάλαμε τον αντιστάτη 220Ω από την στήλη 24 στην στήλη 27 και μετά βάλαμε το LED από την στήλη 27 στην οριζόντια μπλε γραμμή που είναι η γείωση GND. Προσοχή το μεγάλο ποδαράκι του LED ενώνεται με την αντίσταση ενώ το μικρό πάει στην μπλε γραμμή. Φυσικά αντί της στήλης 24 και 27 θα μπορούσαμε να επιλέξουμε όποια θέλουμε αρκεί εκεί να μην υπάρχει άλλη σύνδεση εξαρτημάτων.

Το πρόγραμμα

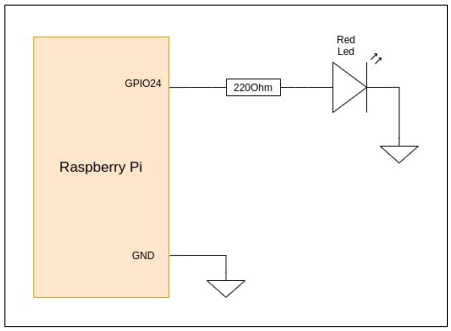
Το πρόγραμμα το γράφουμε στο vscode μέσα από σύνδεση SSH.

```
blink_blocking.py > ...
1   import RPi.GPIO as GPIO
2   import time
3
4   RED_LED = 24 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών
5
6   GPIO.setmode(GPIO.BCM) #Τρόπος αρίθμησης
7   GPIO.setwarnings(False) #Να μην εμφανίζει προειδοποιήσεις
8   GPIO.setup(RED_LED, GPIO.OUT) #Να γίνει έξοδος
9   GPIO.output(RED_LED, GPIO.LOW) #Αρχικά να είναι σβηστό
10
11  #Αναβοσβήνει το LED με περίοδο 1 Second
12  while True: #Για πάντα
13      GPIO.output(RED_LED, GPIO.HIGH) #Άναψε το LED
14      time.sleep(.5) #Καθυστέρησε μισό δευτερόλεπτο
15      GPIO.output(RED_LED, GPIO.LOW) #Σβήσε το LED
16      time.sleep(.5) #Καθυστέρησε μισό δευτερόλεπτο και κάνε το ίδιο από την αρχή για πάντα
17
```

Αντιγράφουμε και επικολλούμε τον κώδικα στο vscode. Προσοχή από το αρχείο pdf δεν μεταφέρονται τα idents, γι’ αυτό πρέπει να τα διορθώσουμε πατώντας το tab.

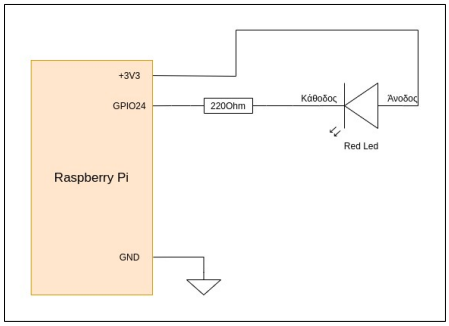
```
import RPi.GPIO as GPIO
import time

RED_LED = 24 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών

GPIO.setmode(GPIO.BCM) #Τρόπος αρίθμησης
GPIO.setwarnings(False) #Να μην εμφανίζει προειδοποιήσεις
GPIO.setup(RED_LED, GPIO.OUT) #Να γίνει έξοδος
GPIO.output(RED_LED, GPIO.LOW) #Αρχικά να είναι σβηστό

#Αναβοσβήνει το LED με περίοδο 1 Second
while True: #Για πάντα
    GPIO.output(RED_LED, GPIO.HIGH) #Άναψε το LED
    time.sleep(.5) #Καθυστέρησε μισό δευτερόλεπτο
    GPIO.output(RED_LED, GPIO.LOW) #Σβήσε το LED
    time.sleep(.5) #Καθυστέρησε μισό δευτερόλεπτο και κάνε το ίδιο από την αρχή για πάντα
```

Το κύκλωμα που μόλις φτιάξαμε είναι το ακόλουθο:

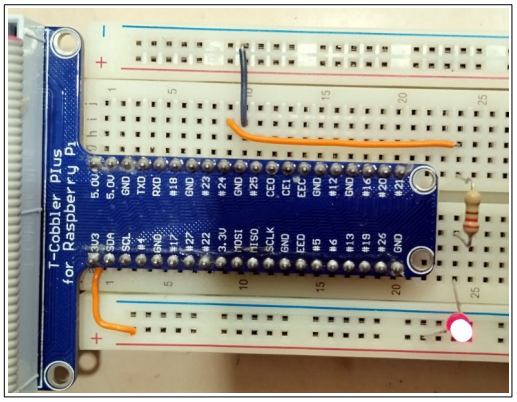


Το LED είναι δίοδος και για να ανάψει πρέπει να είναι ορθά πολωμένο δηλαδή στην άνοδο (μεγάλο ποδαράκι) πρέπει να έχουμε θετικότερη τάση απ’ ότι στην κάθοδο (μικρό ποδαράκι). Με το πρόγραμμα δίνουμε κατά διαστήματα τάση 3,3V (GPIO.HIGH) μέσω του αντιστάτη 220Ω ή 0V (GPIO.LOW). Με το HIGH το LED είναι ορθά πολωμένο και ρέει ρεύμα μέσα απ’ αυτό, επομένως ανάβει για μισό δευτερόλεπτο ενώ με το LOW έχει 0V τόσο στην άνοδο όσο και στην κάθοδο, δηλαδή δεν είναι ορθά πολωμένο και σβήνει για μισό δευτερόλεπτο.

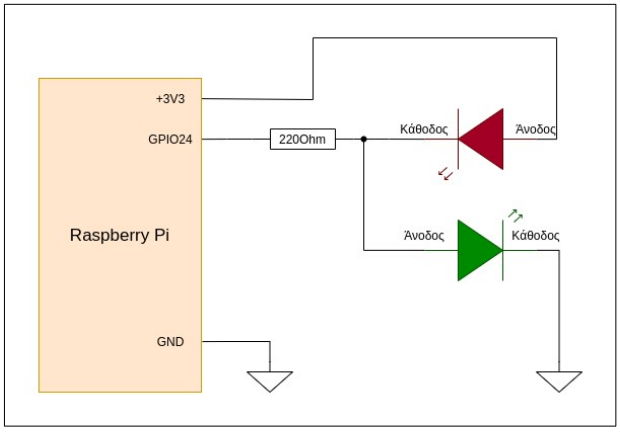
Εναλλακτικά μπορούμε να χρησιμοποιήσουμε το ακόλουθο κύκλωμα:

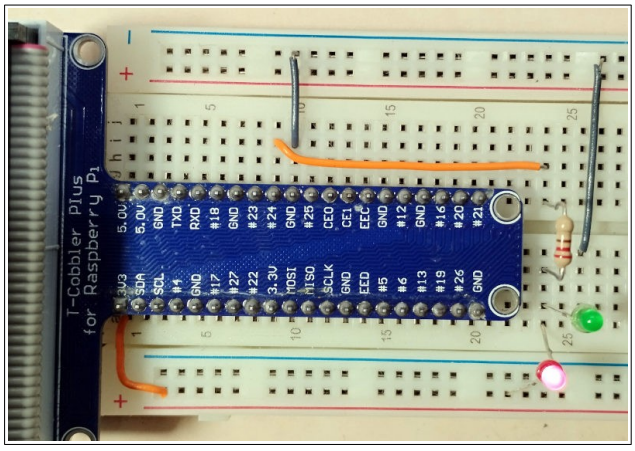


Το νέο κύκλωμα λειτουργεί το ίδιο καλά και το LED αναβοσβήνει με περίοδο 1sec, με την διαφορά ότι τώρα το LED ανάβει όταν το GPIO24 είναι LOW δηλαδή 0V και σβήνει όταν είναι HIGH. Αυτό γίνεται διότι όταν η κάθοδος έχει HIGH 3,3V και η άνοδος είναι μόνιμα στα 3,3V τότε το LED δεν είναι ορθά πολωμένο δηλαδή δεν υπάρχει διαφορά δυναμικού στα άκρα του και δεν ανάβει. Αν τώρα η κάθοδος γίνει LOW 0V τότε η διαφορά δυναμικού είναι 3,3V με θετικότερη την άνοδο δηλαδή ορθά πολωμένο πράγμα που σημαίνει ότι θα ανάψει.
Η υλοποίηση το νέου κυκλώματος φαίνεται στην φωτογραφία. Τώρα δώσαμε τάση +3,3V στην κάτω κόκκινη σειρά και συνδέσαμε την άνοδο του Led (μεγάλο ποδαράκι) σε κάποια οπή αυτής της σειράς.

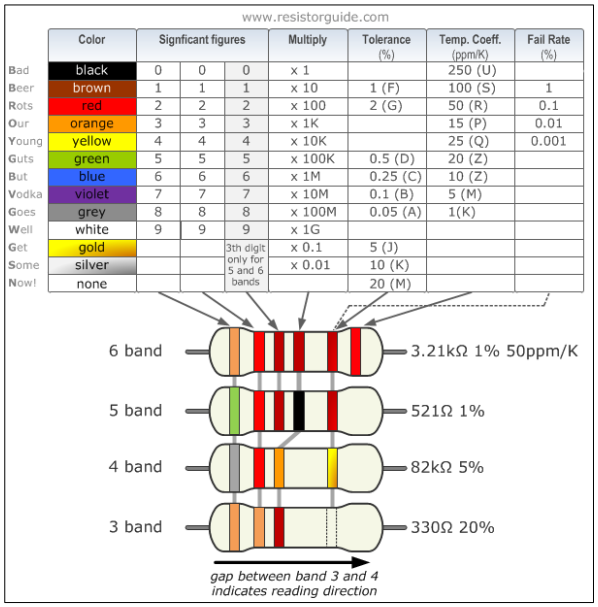


Χρησιμοποιώντας τον ίδιο κώδικα φτιάξτε το παρακάτω κύκλωμα. Τι παρατηρείτε;





Πίνακας χρωμάτων αντιστατών



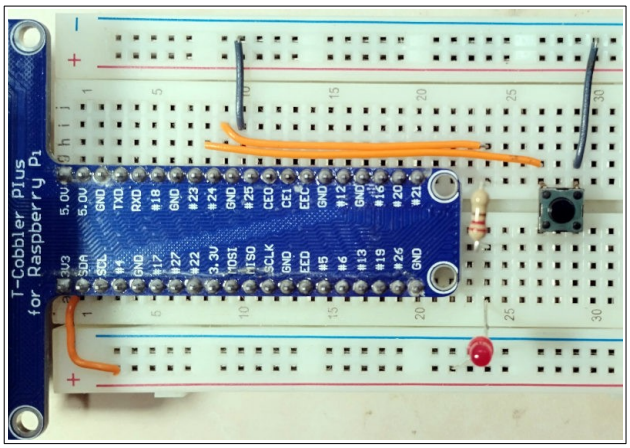
https://abyz.me.uk/rpi/pigpio/python.html

2. Είσοδος με Push Button

Το απλούστερο εξάρτημα για την είσοδο σε ένα σύστημα είναι ο διακόπτης. Ο διακόπτης μπορεί να βρίσκεται μόνο σε δύο καταστάσεις, την κατάσταση ON και την κατάσταση OFF. Επομένως αν συνδέσουμε ένα διακόπτη σε κάποιο pin του GPIO μπορούμε να ανιχνεύσουμε αν είναι στην μία ή στην άλλη κατάσταση. Μια παραλλαγή του απλού διακόπτη είναι το Push Button το οποίο έχει επαναφορά, δηλαδή αν το πατάμε βρίσκεται σε κατάσταση ON και όταν το αφήνουμε επανέρχεται στην κατάσταση OFF.

2.1 Δοκιμή λειτουργίας του Push Button

Συνδέστε ένα Push Button σύμφωνα με την ακόλουθη φωτογραφία.



Συνδέσαμε το GPIO23 με τον ένα ακροδέκτη του Push Button και τον άλλο ακροδέκτη τον οδηγήσαμε στην γείωση GND μέσω την πάνω οριζόντιας μπλε σειράς. Η συνδεσμολογία αυτή είναι γνωστή ως **Pull Up** γιατί εσωτερικά στον controller ενεργοποιείται ένας αντιστάτης >20KΩ ο οποίος συνδέεται στα 3,3V. Όταν το button δεν είναι πατημένο τότε η τάση στο GPIO είναι 3,3V, δηλαδή η τάση περνάει μέσα από τον αντιστάτη και διαβάζουμε λογικό ‘1’. Όταν πιεστεί το button τότε η το GPIO οδηγείται στην γείωση και η τάση πέφτει στα 0V δηλαδή λογικό ‘0’. Με αυτή την συνδεσμολογία όταν πατάμε το button διαβάζουμε ‘0’ ενώ όταν το αφήνουμε διαβάζουμε ‘1’.

Γράψτε στο VS Code τον κώδικα που βρίσκεται ακριβώς από κάτω

button_blocking.py > ...

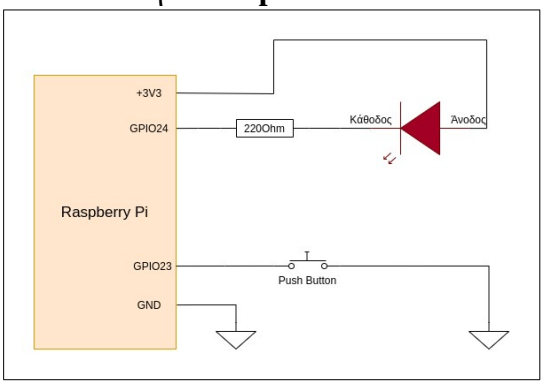
```
import RPi.GPIO as GPIO

BUTTON = 23 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών
GPIO.setmode(GPIO.BCM)  #Τρόπος αρίθμησης
GPIO.setwarnings(False) #Να μην εμφανίζει προειδοποιήσεις
GPIO.setup(BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_UP) #Το GPIO pin είναι είσοδος και σε κατάσταση PULL UP

while True: #Για πάντα
    print(GPIO.input(BUTTON)) #Εμφάνισε την τιμή της εισόδου

```

ή αντιγράψτε από το παρακάτω πλαίσιο. Προσοχή το ident στη while.

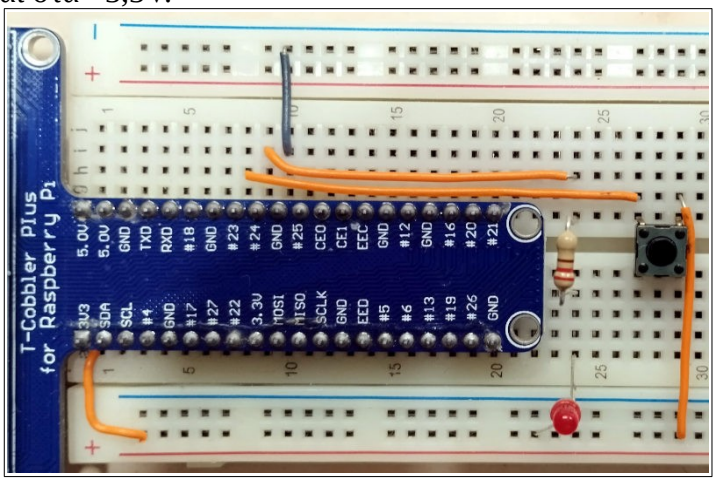
```
import RPi.GPIO as GPIO

BUTTON = 23 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών
GPIO.setmode(GPIO.BCM) #Τρόπος αρίθμησης
GPIO.setwarnings(False) #Να μην εμφανίζει προειδοποιήσεις
GPIO.setup(BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_UP) #Το GPIO pin είναι είσοδος και σε κατάσταση PULL UP

while True: #Για πάντα
    print(GPIO.input(BUTTON)) #Εμφάνισε την τιμή της εισόδου
```

Το πρόγραμμα εμφανίζει συνεχόμενα ‘1’ και όταν πατηθεί το button τότε εμφανίζει συνεχόμενα ‘0’.

Το κύκλωμα του Push Button σε σύνδεση **Pull Up**.

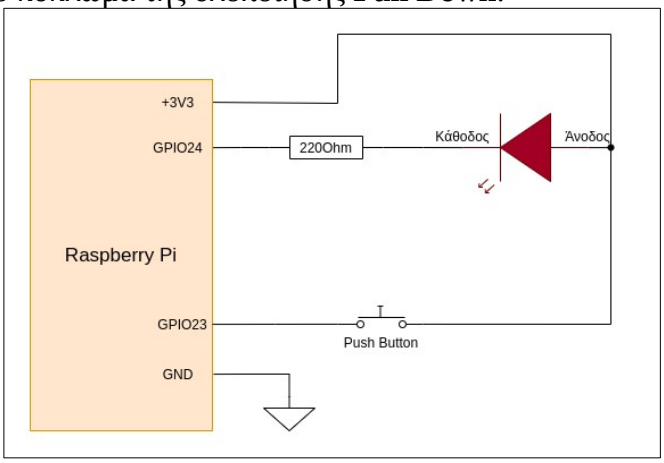


Η δεύτερη εκδοχή είναι η σύνδεση **Pull Down** και τροποποιούμε το κύκλωμα όπως στην παρακάτω φωτογραφία. Τώρα με μια αλλαγή στον κώδικα ενεργοποιούμε τον αντιστάτη pulldown στο εσωτερικό του controller. Στο hardware, ο ακροδέκτης του Push button που πήγαινε πριν στη γείωση, τώρα συνδέεται στα +3,3V.




```
import RPi.GPIO as GPIO

BUTTON = 23 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών
GPIO.setmode(GPIO.BCM) #Τρόπος αρίθμησης
GPIO.setwarnings(False) #Να μην εμφανίζει προειδοποιήσεις
GPIO.setup(BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) #Το GPIO pin είναι είσοδος και σε κατάσταση PULL DOWN

while True: #Για πάντα
    print(GPIO.input(BUTTON)) #Εμφάνισε την τιμή της εισόδου
```

Τώρα το πρόγραμμα εμφανίζει συνεχόμενα ‘0’ και αν πατηθεί το button τότε εμφανίζει ‘1’.
Για να σταματήσουμε την εκτέλεση πατάμε Ctrl + C.
Ακολουθεί το θεωρητικό κύκλωμα της υλοποίησης **Pull Down**.



2.1 Φώτα σκάλας

Χρησιμοποιώντας το προηγούμενο κύκλωμα, να γίνει πρόγραμμα το οποίο θα ελέγχει διαρκώς αν πατήθηκε το Push Button. Αν πατήθηκε θα ανάβει το κόκκινο LED για 10 δευτερόλεπτα και μετά την παρέλευση του χρόνου το LED θα σβήνει.

button_blocking.py > ...

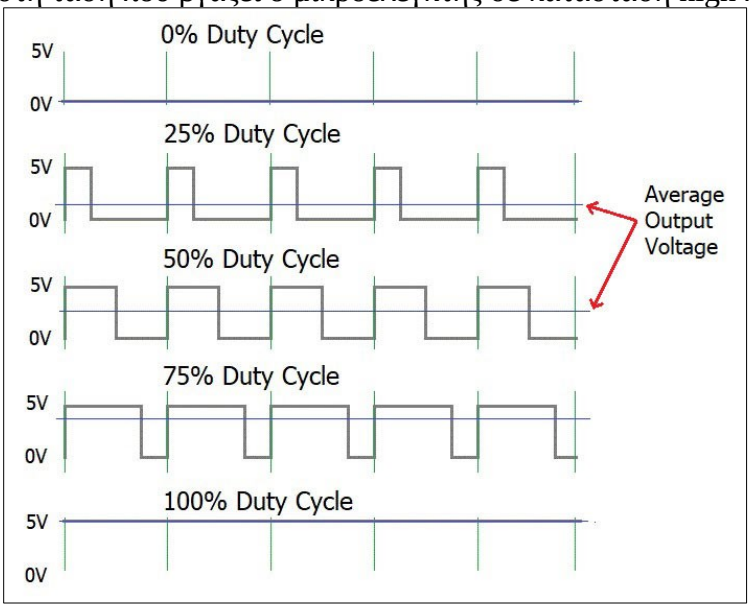
```python
import RPi.GPIO as GPIO
import time

BUTTON = 23 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών
RED_LED = 24 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών

GPIO.setmode(GPIO.BCM)  #Τρόπος αρίθμησης
GPIO.setwarnings(False) #Να μην εμφανίζει προειδοποιήσεις

GPIO.setup(BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) #Το GPIO pin είναι είσοδος και σε κατάσταση PULL DOWN
GPIO.setup(RED_LED, GPIO.OUT) #Να γίνει έξοδος
GPIO.output(RED_LED, GPIO.HIGH) #Αρχικά να είναι σβηστό

while True: #Για πάντα
    #print(GPIO.input(BUTTON)) #(Debug) Εμφάνισε την τιμή της εισόδου
    if GPIO.input(BUTTON): #Αν πατήθηκε το κουμπί
        GPIO.output(RED_LED, GPIO.LOW) #Άναψε το LED
        time.sleep(10) #Περίμενε 10sec
        GPIO.output(RED_LED, GPIO.HIGH) #Σβήσε το LED
    time.sleep(.1) #Καθυστέρηση 100msec μέχρι να ξαναεκτελεστούν οι εντολές
```

Αντιγράψτε και δοκιμάστε το πρόγραμμα. Προσοχή στα idents.

```python
import RPi.GPIO as GPIO
import time

BUTTON = 23 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών
RED_LED = 24 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών

GPIO.setmode(GPIO.BCM) #Τρόπος αρίθμησης
GPIO.setwarnings(False) #Να μην εμφανίζει προειδοποιήσεις

GPIO.setup(BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) #Το GPIO pin είναι είσοδος και σε κατάσταση PULL DOWN
GPIO.setup(RED_LED, GPIO.OUT) #Να γίνει έξοδος
GPIO.output(RED_LED, GPIO.HIGH) #Αρχικά να είναι σβηστό

while True: #Για πάντα
    #print(GPIO.input(BUTTON)) #(Debug) Εμφάνισε την τιμή της εισόδου
    if GPIO.input(BUTTON): #Αν πατήθηκε το κουμπί
        GPIO.output(RED_LED, GPIO.LOW) #Άναψε το LED
        time.sleep(10) #Περίμενε 10sec
        GPIO.output(RED_LED, GPIO.HIGH) #Σβήσε το LED
    time.sleep(.1) #Καθυστέρηση 100msec μέχρι να ξαναεκτελεστούν οι εντολές
```

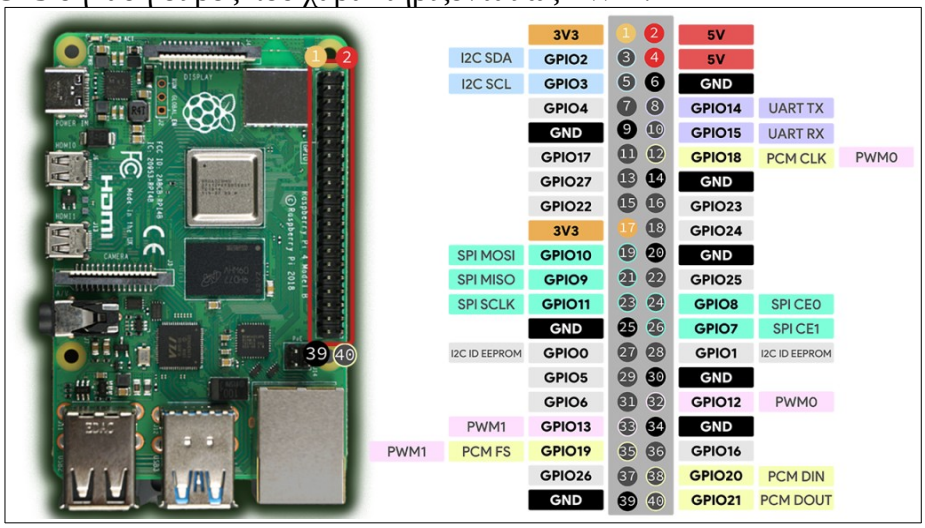
3. Ψευδοαναλογική έξοδος PWM

Μια απλή μέθοδος αναλογικής εξόδου είναι η PWM (Pulse Width Modulation). Με την μέθοδο αυτή οι μικροελεγκτές παράγουν σε κάποιους ακροδέκτες ένα τετραγωνικό παλμό σταθερής συχνότητας μερικών δεκάδων έως και εκατοντάδων Hz (π.χ. 20 - 800Hz). Με κατάλληλες εντολές αλλάζουν την διάρκεια του παλμού. Το σήμα αυτό μετά μπορεί να εξομαλυνθεί με έναν αντιστάτη και ένα πυκνωτή ο οποίος γειώνεται. Στην ένωση αντιστάτη και πυκνωτή έχουμε αναλογικό σήμα από 0V έως την μέγιστη τάση που βγάζει ο μικροελεγκτής σε κατάσταση high π.χ. 3,3V.



Hardware PWM

Το raspberry διαθέτει δύο κανάλια Hardware PWM. Επειδή οι παλμοί παράγονται από κυκλώματα, το hardware PWM έχει μεγάλη σταθερότητα και ακρίβεια και δεν χρειάζεται το software να ασχολείται με την παραγωγή των σημάτων.
Σύμφωνα με το ακόλουθο pinout οι ακροδέκτες με δυνατότητα PWM είναι οι GPIO18, GPIO12, GPIO19, GPIO13 δηλαδή οι ροζ που χαρακτηρίζονται ως PWM.



Αρχικά κάνουμε εγκατάσταση της βιβλιοθήκης με **`sudo pip3 install rpi-hardware-pwm`** . Έπειτα προσθέτουμε στο αρχείο **`/boot/config.txt`** την γραμμή **`dtoverlay=pwm-2chan`** και κάνουμε επανεκκίνηση. Προσοχή αν υπάρχει ενεργή άλλη δήλωση γι' αυτόν τον ακροδέκτη π.χ. dtoverlay=gpio-ir,gpio_pin=18 που είναι για το Lirc τότε πρέπει να απενεργοποιηθεί. Ελέγχουμε με **`lsmod | grep pwm_bcm`** και πρέπει να εμφανίζει κάτι όπως:

```
pwm_bcm2835            16384  1
```

Συνδέστε στο GPIO18 έναν αντιστάτη 220Ω και σε σειρά ένα LED. Η κάθοδος του LED να γειώνεται. Αντιγράψτε και δοκιμάστε τον παρακάτω κώδικα:

pwm_out.py > ...

```
1  from rpi_hardware_pwm import HardwarePWM
2  from time import sleep
3  #Πάντα το chip είναι 0 και βάζουμε συχνότητα 500Hz
4  pwm = HardwarePWM(pwm_channel=0, hz=500, chip=0) #channel 0 για GPIO18 και 1 για GPIO19
5  pwm.start(0) #Ξεκίνα με Duty Cycle 0%
6
7  for i in range(1, 101): #Από 1 έως 100%
8      pwm.change_duty_cycle(i)
9      sleep(.1) #Περίμενε 100 msec
10
11 #Δοκιμή αλλαγής συχνότητας
12 pwm.change_frequency(25_000) #Η 25000 αλλαγή συχνότητας σε 25KHz
13 pwm.change_duty_cycle(50) #50%
14 sleep(5) #Περίμενε 5 δευτερόλεπτα
15 pwm.stop() #Σταμάτα την παραγωγή παλμών PWM
```

Αντιγράψτε και δοκιμάστε το πρόγραμμα. Προσοχή στα idents.

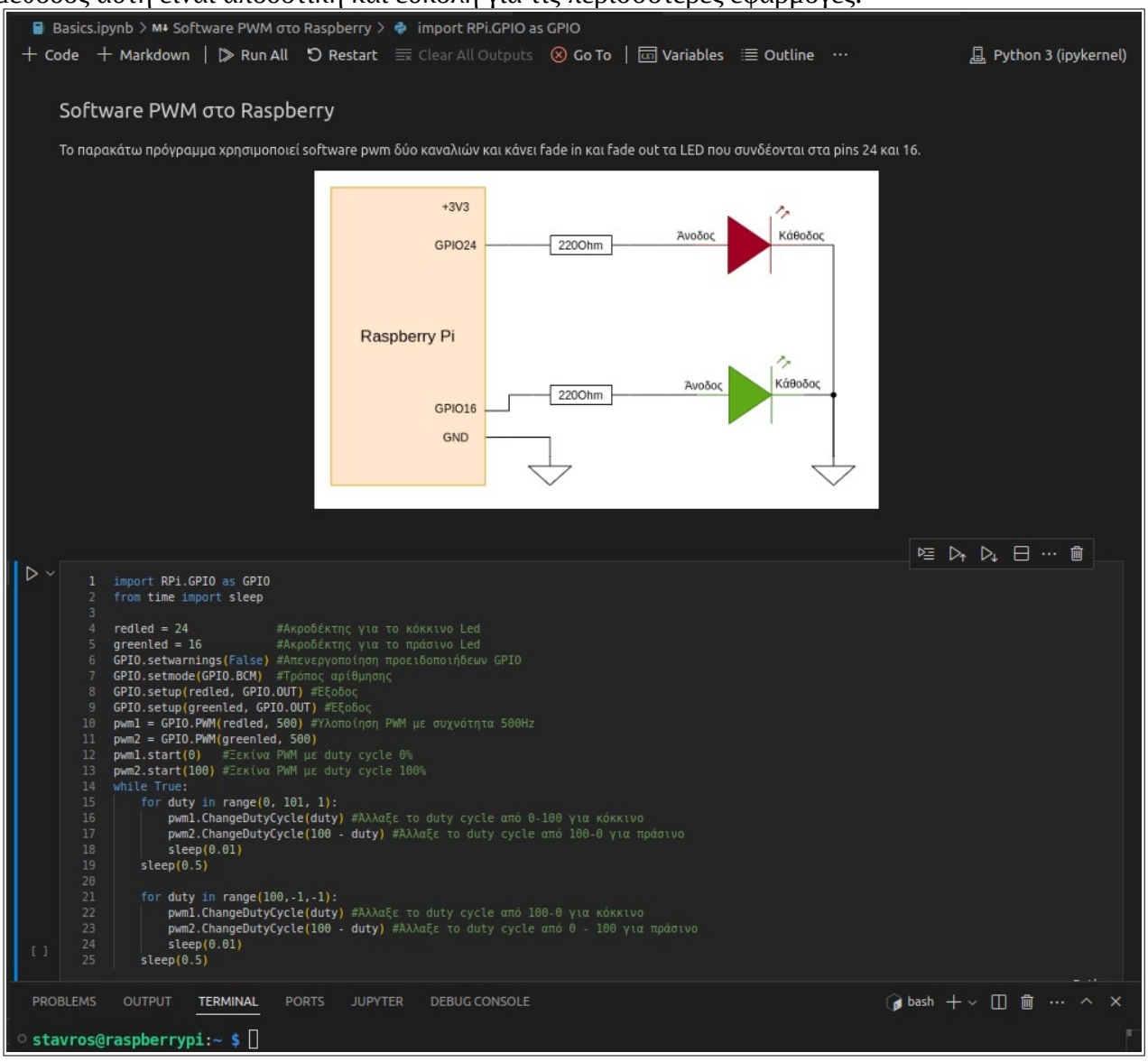
```
import RPi.GPIO as GPIO
from rpi_hardware_pwm import HardwarePWM
from time import sleep
#Πάντα το chip είναι 0 και βάζουμε συχνότητα 500Hz
pwm = HardwarePWM(pwm_channel=0, hz=500, chip=0) #channel 0 για GPIO18 και 1 για GPIO19
pwm.start(0) #Ξεκίνα με Duty Cycle 0%

for i in range(1, 101): #Από 1 έως 100%
    pwm.change_duty_cycle(i)
    sleep(.1) #Περίμενε 100 msec

#Δοκιμή αλλαγής συχνότητας
pwm.change_frequency(25_000) #Η 25000 αλλαγή συχνότητας σε 25KHz
pwm.change_duty_cycle(50) #50%
sleep(5) #Περίμενε 5 δευτερόλεπτα
pwm.stop() #Σταμάτα την παραγωγή παλμών PWM
```

Το πρόγραμμα ανάβει σιγά – σιγά το LED μέχρι το μέγιστο της φωτεινότητας.
Αν θέλουμε το GPIO19 θα βάλουμε pwm_channel=1 στην γραμμή 4.

Για να χρησιμοποιήσουμε τα GPIO12 και GPIO13 γράφουμε στο /boot/config.txt:
`dtoverlay=pwm-2chan,pin=12,func=4,pin2=13,func2=4` και κάνουμε επανεκκίνηση.
Γενικά στο Hardware PWM μπορούμε να χρησιμοποιήσουμε ταυτόχρονα μόνο 2 ακροδέκτες GPIO18, 19 ή GPIO12, 13.

Software PWM

Όλα τα pin του Raspberry μπορούν να χρησιμοποιηθούν για Software PWM. Τώρα η παραγωγή συχνότητας και ο έλεγχος του duty cycle γίνεται από λογισμικό και όχι από υλικό όπως πριν. Η μέθοδος αυτή είναι αποδοτική και εύκολη για τις περισσότερες εφαρμογές.




```
import RPi.GPIO as GPIO
from time import sleep

redled = 24             #Ακροδέκτης για το κόκκινο Led
greenled = 16           #Ακροδέκτης για το πράσινο Led
GPIO.setwarnings(False) #Απενεργοποίηση προειδοποιήσεων GPIO
GPIO.setmode(GPIO.BCM)  #Τρόπος αρίθμησης
GPIO.setup(redled, GPIO.OUT) #Εξοδος
GPIO.setup(greenled, GPIO.OUT) #Εξοδος
pwm1 = GPIO.PWM(redled, 500) #Υλοποίηση PWM με συχνότητα 500Hz
pwm2 = GPIO.PWM(greenled, 500)
pwm1.start(0)   #Ξεκίνα PWM με duty cycle 0%
pwm2.start(100) #Ξεκίνα PWM με duty cycle 100%
while True:
    for duty in range(0, 101, 1):
        pwm1.ChangeDutyCycle(duty) #Άλλαξε το duty cycle από 0-100 για κόκκινο
        pwm2.ChangeDutyCycle(100 - duty) #Άλλαξε το duty cycle από 100-0 για πράσινο
        sleep(0.01)
    sleep(0.5)

    for duty in range(100,-1,-1):
        pwm1.ChangeDutyCycle(duty) #Άλλαξε το duty cycle από 100-0 για κόκκινο
        pwm2.ChangeDutyCycle(100 - duty) #Άλλαξε το duty cycle από 0 - 100 για πράσινο
        sleep(0.01)
    sleep(0.5)
```

Αντιγράψτε τον κώδικα και δοκιμάστε τον. Προσοχή στις εσοχές.

```
import RPi.GPIO as GPIO
from time import sleep

redled = 24 #Ακροδέκτης για το κόκκινο Led
greenled = 16 #Ακροδέκτης για το πράσινο Led
GPIO.setwarnings(False) #Απενεργοποίηση προειδοποιήσεων GPIO
GPIO.setmode(GPIO.BCM) #Τρόπος αρίθμησης
GPIO.setup(redled, GPIO.OUT) #Έξοδος
GPIO.setup(greenled, GPIO.OUT) #Έξοδος
pwm1 = GPIO.PWM(redled, 500) #Υλοποίηση PWM με συχνότητα 500Hz
pwm2 = GPIO.PWM(greenled, 500)
pwm1.start(0) #Ξεκίνα PWM με duty cycle 0%
pwm2.start(100) #Ξεκίνα PWM με duty cycle 100%
while True:
    for duty in range(0, 101, 1):
        pwm1.ChangeDutyCycle(duty) #Άλλαξε το duty cycle από 0-100 για κόκκινο
        pwm2.ChangeDutyCycle(100 - duty) #Άλλαξε το duty cycle από 100-0 για πράσινο
```

```
        sleep(0.01)
    sleep(0.5)

    for duty in range(100,-1,-1):
        pwm1.ChangeDutyCycle(duty) #Άλλαξε το duty cycle από 100-0 για κόκκινο
        pwm2.ChangeDutyCycle(100 - duty) #Άλλαξε το duty cycle από 0 - 100 για πράσινο
        sleep(0.01)
    sleep(0.5)
```

Jupyter σε vscode

Στο vscode έχουμε την δυνατότητα να χρησιμοποιούμε σημειωματάρια (notebooks) του jupyter. Αρχικά εγκαθιστούμε το jupyter lab στο raspberry.

```
sudo python3 -m pip install --upgrade pip
```

Στο raspberry εγκαθιστώ το jupyter και jupyterlab.

```
pip3 install jupyter
pip3 install jupyterlab
```

Κάνει εγκατάσταση στο `~/.local/bin`

Καλύτερα σαν root σε όλο το σύστημα και όχι στον χρήστη. Επομένως τα πορτοκαλί παραλείπονται. Κάνω εγκατάσταση:

```
sudo pip3 install jupyterlab
```

Τα αρχεία πάνε στο `/usr/local/bin`

`cd ~/.local/bin` Αν είναι τοπικά και όχι σε όλο το σύστημα, διαφορετικά είναι στο path.

Φτιάχνω configuration file με:

```
./jupyter-lab --generate-config
```

Το αρχείο υπάρχει στο `~/.jupyter` και ονομάζεται `jupyter_lab_config.py`

Φτιάχνω hashed password με:

```
jupyter notebook password
```

Αυτό αποθηκεύεται στο `~/.jupyter/jupyter_server_config.json` και το αντιγράφω στο `jupyter_lab_config.py` στην θέση `c.ServerApp.password = …….`

Διορθώνω το `jupyter_lab_config.py` και συγκεκριμένα ενεργοποιώ τις γραμμές:

```
c.ServerApp.ip = '0.0.0.0'
c.ServerApp.notebook_dir = '/home/stavros'
c.ServerApp.open_browser = False
c.ServerApp.password = 'argon2:$argon2id$v=19$m=10240,t=10,p=8$OVwZst...….'
c.ServerApp.port = 8888
```

Δοκιμάζω τον server με

```
/usr/bin/python3 /usr/local/bin/jupyter-lab --config /home/stavros/.jupyter/jupyter_lab_config.py
```

Εγκατάσταση service

```
sudo nano /lib/systemd/system/jupyter.service
```

```
[Unit]
Description=Jupyter Lab Server

[Service]
Type=simple
PIDFile=/run/jupyter.pid

#EnvironmentFile=/home/fermi/.jupyter/env

ExecStart=/usr/local/bin/jupyter-lab --config=/home/stavros/.jupyter/jupyter_notebook_config.py

User=stavros
Group=stavros
WorkingDirectory=/home/stavros
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

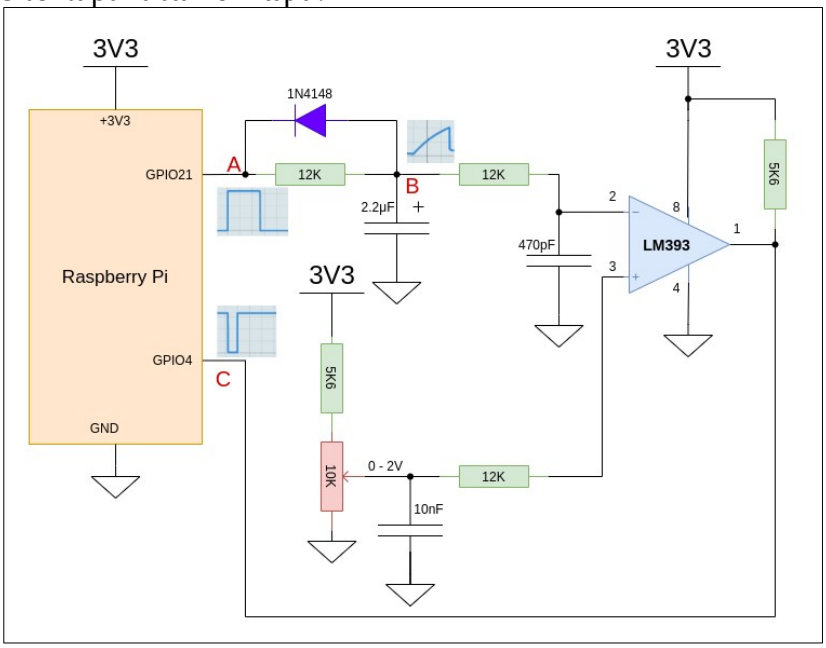
```
sudo systemctl enable jupyter.service
sudo systemctl start jupyter.service
```

Στο vscode στον υπολογιστή βάζω extensions jupyter στο raspberry και πολλά άλλα. Ανοίγω ένα notebook και πατάω δεξιά στον πυρήνα. Connect to jupyter server και δίνω

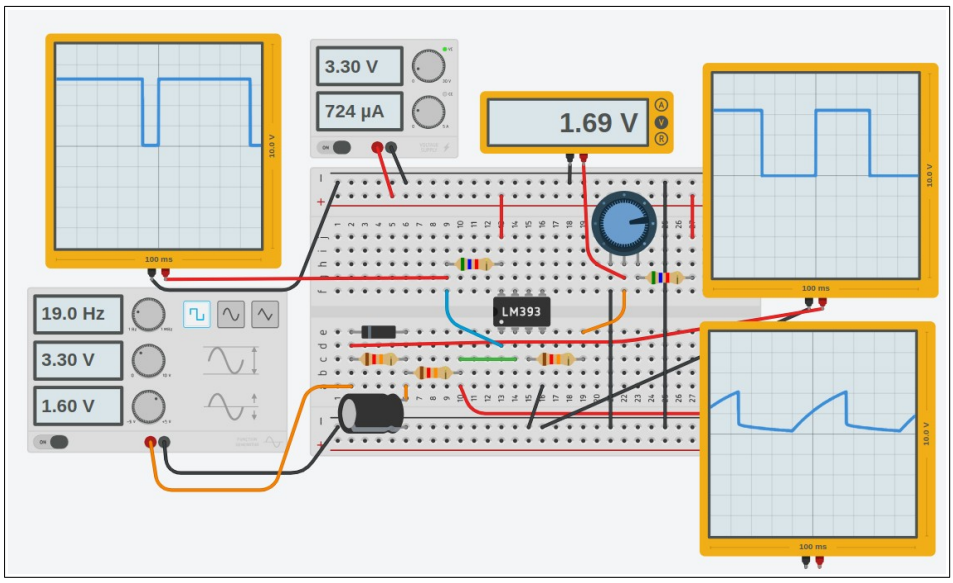
http://192.168.42.122:8888 και μετά ζητάει password και μετά πυρήνα στο απομακρυσμένο σύστημα. Τέλος δίνω ένα φιλικό όνομα στον raspberry server.

4. Αναλογική είσοδος

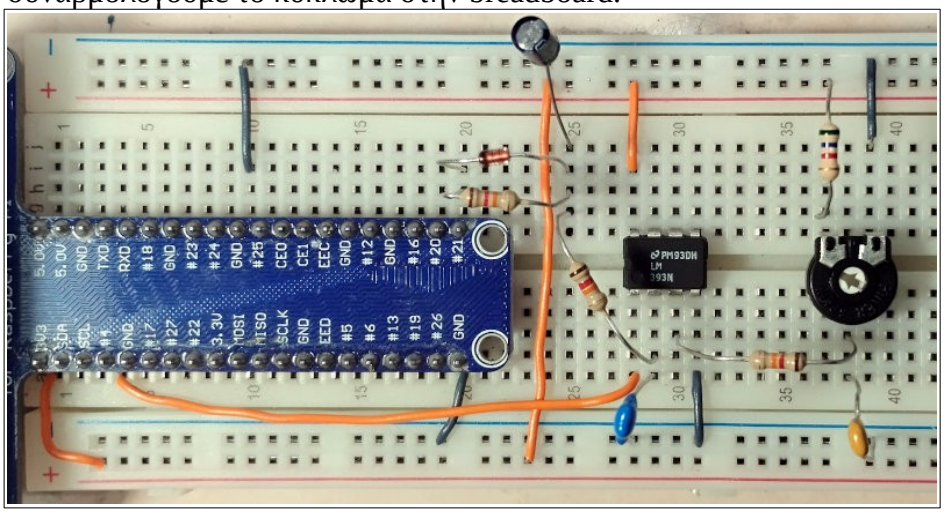
Επειδή τα πάντα στην φύση είναι αναλογικά, απαιτείται πολλές φορές οι υπολογιστές να δέχονται αναλογικά σήματα όπως σήματα ήχου, μεταβολές τάσης κλπ. Δυστυχώς το Raspberry δεν διαθέτει κανέναν μετατροπέα από αναλογικό σε ψηφιακό (ADC). Αν θέλουμε να διαβάσουμε αναλογικά σήματα τότε πρέπει να συνδέσουμε μέσω κάποιου διαύλου SPI ή I2C ένα chip ADC ή κάποια πλακέτα Arduino η οποία διαθέτει αρκετά αναλογικά κανάλια.
Αν όμως δεν θέλουμε μεγάλη ταχύτητα δειγματοληψίας και μεγάλη ευκρίνεια τότε μπορούμε να χρησιμοποιήσουμε το παρακάτω κύκλωμα.



Εδώ χρησιμοποιούμε δύο pins GPIO του Raspberry. Το 21 είναι έξοδος και παράγει ασύγχρονα έναν παλμό 27msec. Η τιμή αυτή υπολογίζεται από τον τύπο T = R*C δηλαδή $12*10^{3} * 2,2*10^{-6}$ το οποίο δίνει 26,4msec. Στον χρόνο αυτό ο πυκνωτής φορτίζεται κατά το 63,2% της τάσης δηλαδή 3,3V * 0,63 = 2V. Στον χρόνο αυτό η καμπύλη φόρτισης είναι αρκετά γραμμική γι' αυτό ο μετατροπέας μας θα λειτουργεί για τάσεις από 0 έως 2V.
Ο πυκνωτής 2,2μF φορτίζεται μέσα από τον αντιστάτη 12K με την άνοδο του παλμού και η ανερχόμενη τάση συνδέεται στην αντιστρέφουσα είσοδο (-) του συγκριτή. Με την άνοδο αρχίζουμε να μετράμε τον χρόνο. Στην μη αντιστρέφουσα είσοδο (+) συνδέεται η προς μέτρηση τάση από το ποτενσιόμετρο 10K. Όσο η τάση στο (-) είναι χαμηλότερη στην έξοδο έχουμε λογικό '1'. Μόλις η τάση στο (-) ξεπεράσει για λίγο την τάση προς μέτρηση στο (+) τότε στην έξοδο θα έχουμε λογικό '0'. Αυτή η πτώση του παλμού ανιχνεύεται ως συμβάν από το pin 4 που είναι είσοδος. Τότε μετράμε τον χρόνο από την άνοδο του παλμού και σταματάμε τον παλμό. Ο πυκνωτής εκφορτίζεται γρήγορα μέσα από την δίοδο 1N4148. Δηλαδή όσο πιο μεγάλη είναι η τάση θα μετράμε περισσότερο χρόνο από την άνοδο του παλμού στο 21.
Αρχικά κάνουμε μια προσομοίωση του κυκλώματος στο Tinkercad για να δούμε την συμπεριφορά του.



Στη συνέχεια συναρμολογούμε το κύκλωμα στην breadboard.



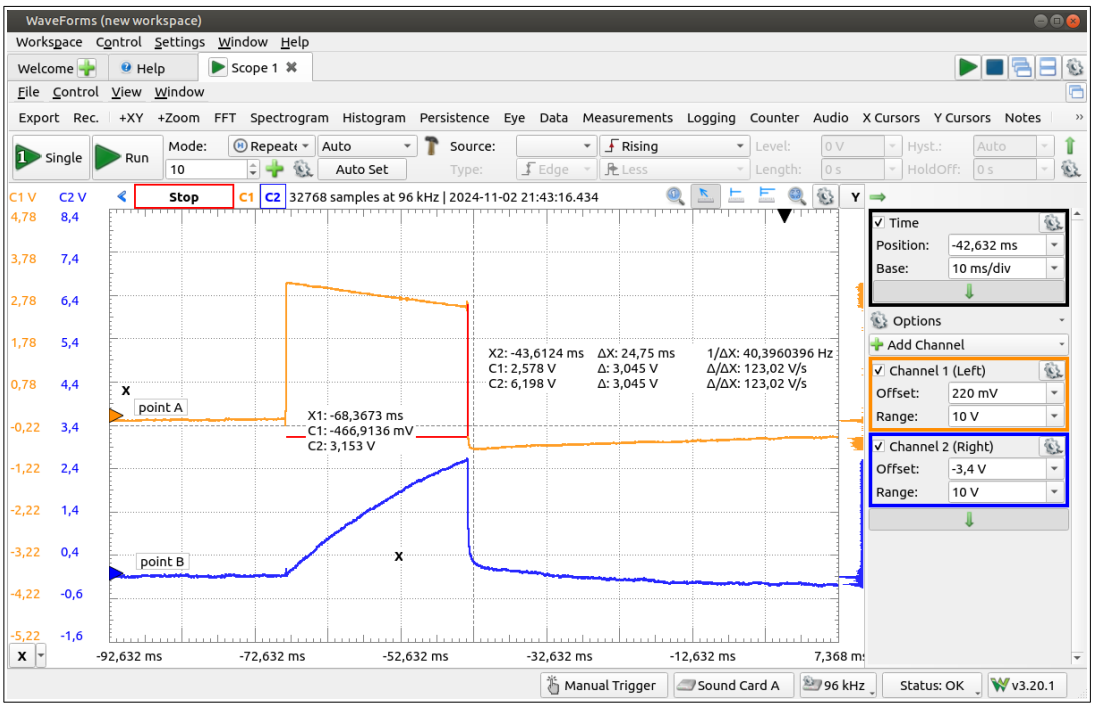
Γράφουμε τον κώδικα σε python 3.

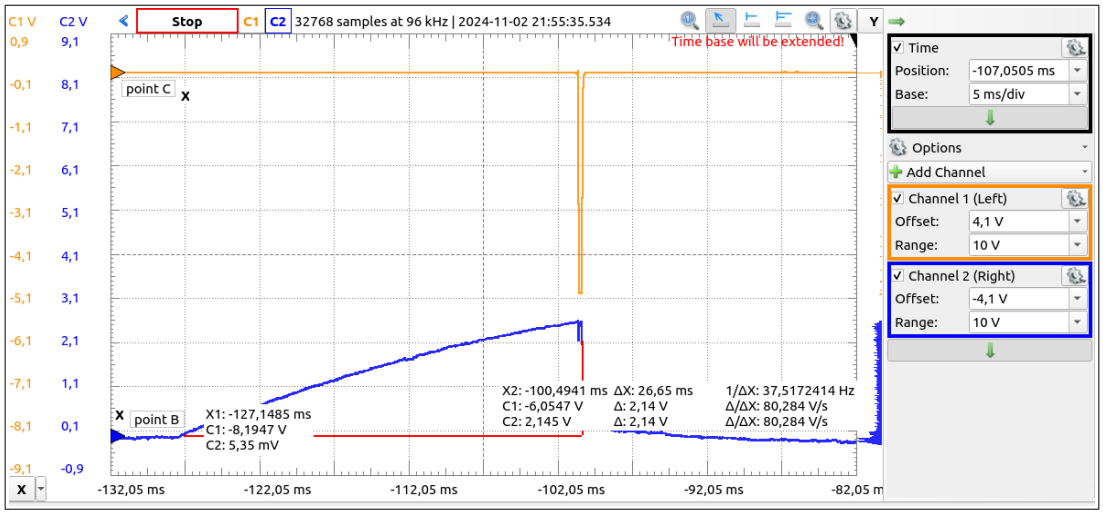
adc_conv1.py > ...

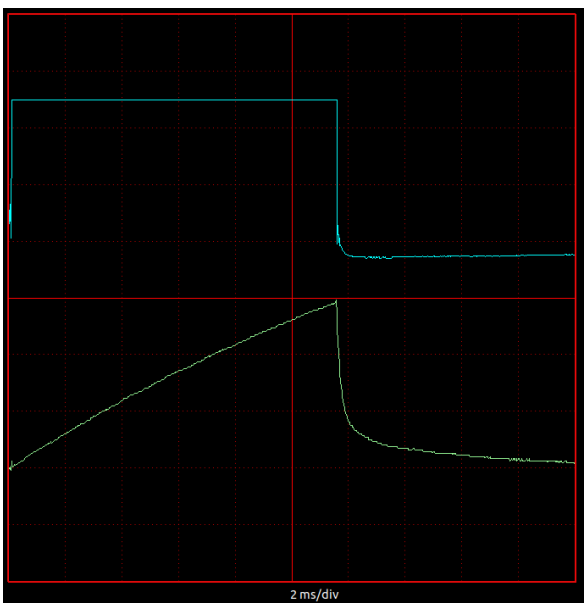
```python
import RPi.GPIO as GPIO
import time, threading

def generateVref():
    global t1, PulseRaise
    while Running:
        GPIO.output(REF_PIN, GPIO.HIGH) #Φόρτιση πυκνωτή
        PulseRaise = True
        t1 = time.perf_counter() #Πάρε το χρόνο τώρα κατά την άνοδο του παλμού
        time.sleep(.027) #T=R*C = 12K * 2u2 = 26.4msec για το 0,632 63,2% της τάσης δηλαδή 1T. 3,3 x .632 = 2.08V
        GPIO.output(REF_PIN, GPIO.LOW) #Εκφόρτιση πυκνωτή. Χρειάζεται σε περίπτωση μικρής τιμής πυκνωτή
        time.sleep(.2) #Περίπου 5 δείγματα / sec

def getADC(gpio):
    global dt, PulseRaise
    if PulseRaise: #Μόνο αν έχει ξεκινήσει η άνοδος του παλμού να μετρήσει τον χρόνο για να αποφύγει πλασματικά interrupt
        t2 = time.perf_counter()
        GPIO.output(REF_PIN, GPIO.LOW) #Εκφόρτιση πυκνωτή
        dt = t2 - t1
        PulseRaise = False

#Πίνακες διόρθωσης μη γραμμικής συμπεριφοράς της φόρτισης του πυκνωτή
volts = [0.0, .1, .2, .3, .4, .5, .6, .7, .8, .9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1]
vals  = [  0,  8, 16, 25, 34, 44, 53, 64, 74, 85,  97, 109, 121, 135, 149, 164, 179, 196, 215, 234, 251, 252] #Αλλάξτε μόνο εδώ

def readADC():
    #print(dt) #.003 - .026 sec. Βάζω μέσα αυτή την εντολή για να δω το εύρος του dt από 0 έως 2V
    offs = 2  #Αρχικά το κάνω 0 και βλέπω την τιμή της val για είσοδο 0V. (Συνήθως 2-3)
    val = int(dt * 10000) - offs  #Μετατροπή τιμών σε ακέραιες. Τιμές από 0 - 252
    #print(val) #Debug
    if val < 0:
        val = 0
    if val > 252:
        val = 252
    #print(val) #Debug
    #--- Για να φτιάξω τους πίνακες βάζω σχόλια στις παρακάτω γραμμές
    #''' <-- Βγάλε # για διόρθωση μη γραμμικής συμπεριφοράς
    p = 0
    found = False
    while p < len(vals) and not found:
        if val > vals[p]:
            p += 1
        else:
            found = True
    if p != 0:
        v = volts[p-1] + (.1 * (val - vals[p-1])) / (vals[p] - vals[p-1])
    else:
        v = 0
    return round(v, 2)
    #''' #<-- Βγάλε # στην αρχή για διόρθωση μη γραμμικής συμπεριφοράς
    #--- Σχόλια μέχρι εδώ και ενεργοποιώ την επόμενη. Αυξάνω την τάση ανά .1V με βολτόμετρο αναφοράς και διαβάζω την τιμή του val
    #και διορθώνω τον πίνακα vals
    print(val) #Debug
```

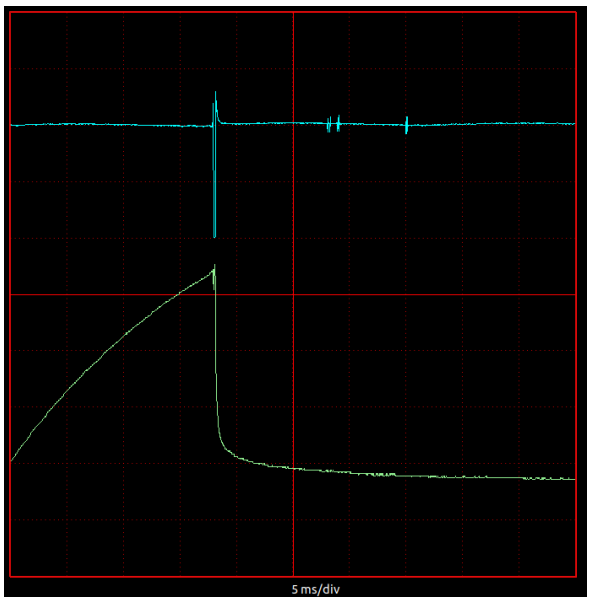
```python

# =======  Κυρίως πρόγραμμα ===========================================================
REF_PIN = 21            #Ακροδέκτης για παραγωγή της τάσης αναφοράς
COMP_PIN = 4            #Ακροδέκτης εισόδου από τον συγκριτή
GPIO.setwarnings(False) #Απενεργοποίηση προειδοποιήδεων GPIO
GPIO.setmode(GPIO.BCM)  #Τρόπος αρίθμησης
GPIO.setup(REF_PIN, GPIO.OUT)  #Ψηφιακή έξοδος
GPIO.output(REF_PIN, GPIO.LOW) #Αρχικά βγάλε λογικό 0
GPIO.setup(COMP_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP) #Είσοδος Pull Up
GPIO.add_event_detect(COMP_PIN, GPIO.FALLING, callback = getADC) #Προσθήκη συμβάντος ώστε να γίνεται διακοπή με κάθοδο παλμού
t1 = dt = 0
PulseRaise = False
Running = True #Flag που δηλώνει αν τρέχει η εφαρμογή
t = threading.Thread(target = generateVref) #Νέο thread για παραγωγή Vref από τετραγωνικό παλμό
t.start() #Ξεκινάει το thread

try:
    while True: #Για πάντα
        time.sleep(1)
        print(readADC(), "V")

except KeyboardInterrupt:
    Running = False
    t.join()
```

Στο πρόγραμμα Waveforms ή το xoscope που είναι ψηφιακοί παλμογράφοι υπολογιστή για την κάρτα ήχου, ελέγχουμε την σωστή λειτουργία του κυκλώματος. Τα σημεία A, B, C είναι αυτά που φαίνονται στο θεωρητικό κύκλωμα.


WaveForms (new workspace)
Workspace Control Settings Window Help
Welcome Help Scope 1
File Control View Window
Export Rec. +XY +Zoom FFT Spectrogram Histogram Persistence Eye Data Measurements Logging Counter Audio X Cursors Y Cursors Notes
Single Run Mode: Repeat Auto Source: Rising
Stop C1 C2 32768 samples at 96 kHz | 2024-11-02 21:43:16.434
point A
point B
X1: -68,3673 ms
C1: -466,9136 mV
C2: 3,153 V
X2: -43,6124 ms ΔX: 24,75 ms 1/ΔX: 40,3960396 Hz
C1: 2,578 V Δ: 3,045 V Δ/ΔX: 123,02 V/s
C2: 6,198 V Δ: 3,045 V Δ/ΔX: 123,02 V/s
Time
Position: -42,632 ms
Base: 10 ms/div
Options
Add Channel
Channel 1 (Left)
Offset: 220 mV
Range: 10 V
Channel 2 (Right)
Offset: -3,4 V
Range: 10 V
-92,632 ms -72,632 ms -52,632 ms -32,632 ms -12,632 ms 7,368 ms
Manual Trigger Sound Card A 96 kHz Status: OK v3.20.1


Stop C1 C2 32768 samples at 96 kHz | 2024-11-02 21:55:35.534
Time base will be extended!
point C
point B
X1: -127,1485 ms
C1: -8,1947 V
C2: 5,35 mV
X2: -100,4941 ms ΔX: 26,65 ms 1/ΔX: 37,5172414 Hz
C1: -6,0547 V Δ: 2,14 V Δ/ΔX: 80,284 V/s
C2: 2,145 V Δ: 2,14 V Δ/ΔX: 80,284 V/s
Time
Position: -107,0505 ms
Base: 5 ms/div
Options
Add Channel
Channel 1 (Left)
Offset: 4,1 V
Range: 10 V
Channel 2 (Right)
Offset: -4,1 V
Range: 10 V
-132,05 ms -122,05 ms -112,05 ms -102,05 ms -92,05 ms -82,05 ms


2 ms/div


5 ms/div

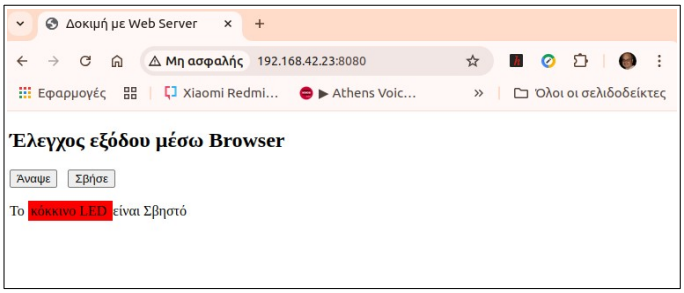
Ακολουθεί ο κώδικας σε μορφή κειμένου.

```
import RPi.GPIO as GPIO
import time, threading

def generateVref():
    global t1, PulseRaise
    while Running:
        GPIO.output(REF_PIN, GPIO.HIGH) #Φόρτιση πυκνωτή
        PulseRaise = True
        t1 = time.perf_counter() #Πάρε το χρόνο τώρα κατά την άνοδο του παλμού
        time.sleep(.027) #T=R*C = 12K * 2u2 = 26.4msec για το 0,632 63,2% της τάσης δηλαδή 1T. 3,3 x .632 = 2.08V
        GPIO.output(REF_PIN, GPIO.LOW) #Εκφόρτιση πυκνωτή. Χρειάζεται σε περίπτωση μικρής τιμής πυκνωτή
        time.sleep(.2) #Περίπου 5 δείγματα / sec

def getADC(gpio):
    global dt, PulseRaise
    if PulseRaise: #Μόνο αν έχει ξεκινήσει η άνοδος του παλμού να μετρήσει τον χρόνο για να αποφύγει πλασματικά interrupt
        t2 = time.perf_counter()
        GPIO.output(REF_PIN, GPIO.LOW) #Εκφόρτιση πυκνωτή
        dt = t2 - t1
        PulseRaise = False

#Πίνακες διόρθωσης μη γραμμικής συμπεριφοράς της φόρτισης του πυκνωτή
volts = [0.0, .1, .2, .3, .4, .5, .6, .7, .8, .9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1]
vals = [ 0, 8, 16, 25, 34, 44, 53, 64, 74, 85, 97, 109, 121, 135, 149, 164, 179, 196, 215, 234, 251, 252] #Αλλάξτε μόνο εδώ

def readADC():
    #print(dt) #.003 - .026 sec. Βάζω μέσα αυτή την εντολή για να δω το εύρος του dt από 0 έως 2V
    offs = 2 #Αρχικά το χάνω 0 και βλέπω την τιμή της val για είσοδο 0V. (Συνήθως 2-3)
    val = int(dt * 10000) - offs #Μετατροπή τιμών σε ακέραιες. Τιμές από 0 - 252
    #print(val) #Debug
    if val < 0:
        val = 0
    if val > 252:
        val = 252
    #print(val) #Debug
    #--- Για να φτιάξω τους πίνακες βάζω σχόλια στις παραχάτω γραμμές
    #''' <-- Βγάλε # για διόρθωση μη γραμμικής συμπεριφοράς
    p = 0
    found = False
    while p < len(vals) and not found:
        if val > vals[p]:
            p += 1
        else:
            found = True
    if p != 0:
        v = volts[p-1] + (.1 * (val - vals[p-1])) / (vals[p] - vals[p-1])
    else:
        v = 0
    return round(v, 2)
    #''' #<-- Βγάλε # στην αρχή για διόρθωση μη γραμμικής συμπεριφοράς
    #--- Σχόλια μέχρι εδώ και ενεργοποιώ την επόμενη. Αυξάνω την τάση ανά .1V με βολτόμετρο αναφοράς και διαβάζω την τιμή του val
    #και διορθώνω τον πίνακα vals
    print(val) #Debug

# ======= Κυρίως πρόγραμμα =================================================================
REF_PIN = 21 #Ακροδέκτης για παραγωγή της τάσης αναφοράς
COMP_PIN = 4 #Ακροδέκτης εισόδου από τον συγκριτή
GPIO.setwarnings(False) #Απενεργοποίηση προειδοποιήσεων GPIO
GPIO.setmode(GPIO.BCM) #Τρόπος αρίθμησης
GPIO.setup(REF_PIN, GPIO.OUT) #Ψηφιακή έξοδος
GPIO.output(REF_PIN, GPIO.LOW) #Αρχικά βγάλε λογικό 0
GPIO.setup(COMP_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP) #Είσοδος Pull Up
GPIO.add_event_detect(COMP_PIN, GPIO.FALLING, callback = getADC) #Προσθήκη συμβάντος ώστε να γίνεται διακοπή με χάθοδο παλμού
t1 = dt = 0
PulseRaise = False
Running = True #Flag που δηλώνει αν τρέχει η εφαρμογή
t = threading.Thread(target = generateVref) #Νέο thread για παραγωγή Vref από τετραγωνικό παλμό
t.start() #Ξεκινάει το thread

try:
    while True: #Για πάντα
        time.sleep(1)
        print(readADC(), "V")
except KeyboardInterrupt:
    Running = False
    t.join()
```

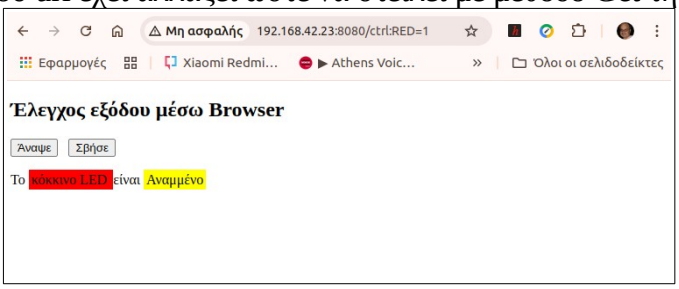
Να επισημάνουμε ότι η παραγωγή της τάσης αναφοράς δηλαδή η φόρτιση του πυκνωτή γίνεται από την συνάρτηση generateVref η οποία τρέχει συνεχώς σε ξεχωριστό **thread** και φορτίζει - εκφορτίζει τον πυκνωτή με συχνότητα πέντε φορές το δευτερόλεπτο. Κατά την έναρξη της φόρτισης δηλαδή όταν αρχίσει η άνοδος του παλμού στο REF_PIN, κρατάει τον χρόνο t1 σε ns. Αν ο μέγιστος χρόνος παρέλθει χωρίς να γίνει interrupt τότε γίνεται κατέβασμα παλμού στο REF_PIN, ώστε να εκφορτιστεί ο πυκνωτής. Αν γίνει διακοπή τότε καλείται η συνάρτηση εξυπηρέτησης διακοπής getADC. Εδώ αφαιρείται ο τρέχων χρόνος από τον t1 και υπολογίζεται ο dt που είναι ο χρόνος ο οποίος χρειάστηκε για να φτάσει η τάση του πυκνωτή την προς μέτρηση τάση. Οι λίστες volts και vals χρησιμοποιούνται για να διορθωθεί η μη γραμμική συμπεριφορά της καμπύλης φόρτισης του πυκνωτή.

5. Έλεγχος εξόδου μέσω browser

Στο πρόγραμμα που ακολουθεί θα χρησιμοποιήσουμε τον ενσωματωμένο web server της Python για να φτιάξουμε μια απλή σελίδα διεπαφής ώστε ο χρήστης ν' ανάβει ή να σβήνει ένα LED. Στην σελίδα υπάρχουν δύο buttons για τον έλεγχο του led και μια γραμμή κειμένου που μας ενημερώνει για την τρέχουσα κατάσταση του LED.
Αρχικά γράφω στο url την τοπική διεύθυνση του raspberry και πόρτα 8080 και εμφανίζεται η παρακάτω σελίδα.



Αν πατήσω το button '***Άναψε***', θα ανάψει το led στο breadboard και η σελίδα θα αλλάξει. Παρατηρήστε το path του url έχει αλλάξει ώστε να στείλει με μέθοδο Get την επιλογή του χρήστη.



Τέλος αν πατήσω το button '***Σβήσε***', θα σβήσει το led και η σελίδα θα γίνει όπως φαίνεται παρακάτω.

Παρατηρήστε την αλλαγή στο path του url.

Ακολουθεί ο κώδικας όπως φαίνεται στο vscode.

webGUI.py > ...

```python
#nmap -sP 192.168.42.0/24 στο Linux να βρω την IP του Raspberry
from http.server import BaseHTTPRequestHandler, HTTPServer
import time, socket
import RPi.GPIO as GPIO
from threading import Thread

RED_LED = 18 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών

#Η σελίδα που θα προβληθεί σε HTML και λίγο Java Script
#Όπου υπάρχει %s θα αντικατασταθεί με τις παραμέτρους κατά την αποστολή της σελίδας
p1='''<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
 <title>Δοκιμή με Web Server</title>
 <script type="text/javascript">
   function set_RED(s)
    {
     if (s == 1)
        location.href = "/ctrl:RED=1";
     else if (s == 0)
        location.href = "/ctrl:RED=0";
    }
 </script>
</head>
<body>
 <H2>Έλεγχος εξόδου μέσω Browser</H2>
 <button type="button" id="RED_ON" onclick="set_RED(1)">Άναψε</button>  
 <button type="button" id="RED_OFF" onclick="set_RED(0)">Σβήσε</button>  <br><br>
 Το <span style="background-color:red; padding:3px;"> κόκκινο LED </span> είναι %s
</body>
</html>'''

state = "Σβηστό" #Αρχικά η κατάσταση του LED είναι σβηστό

#Ορισμός κλάσης του Web Server
class MyServer(BaseHTTPRequestHandler):
    def do_GET(self): #Αν υπάρχει αίτημα GET
        global state
        #Αν ζητήται η αρχική σελίδα χωρίς παραμέτρους
        if self.path == '/':
            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()
            self.wfile.write(bytes(p1 %(state), "utf-8")) #Στείλε και αντικατέστησε την μία παράμετρο με την μεταβλητή state
        #Αν ζητήται με την διεύθυνση /ctrl
        elif '/ctrl' in self.path:
            cmd = self.path.split(":") #Χώρισε το path βάσει της ':'
            #Αν υπάρχει το get /ctrl:RED=1 θα ανάψει
            if "RED=1" in cmd[1]:
                state = '<span style="background-color:yellow; padding:3px;"> Αναμμένο </span>'
                GPIO.output(RED_LED, GPIO.HIGH) #Άναψε το LED
                self.send_response(200)
                self.send_header("Content-type", "text/html")
                self.end_headers()
                self.wfile.write(bytes(p1 %(state), "utf-8"))
            #Αν υπάρχει το get /ctrl:RED=0 θα σβήσει
            if "RED=0" in cmd[1]:
                state = '<span style="background-color:black; color:white; padding:3px;"> Σβηστό </span>'
                GPIO.output(RED_LED, GPIO.LOW) #Σβήσε το LED
                self.send_response(200)
                self.send_header("Content-type", "text/html")
                self.end_headers()
                self.wfile.write(bytes(p1 %(state), "utf-8"))

#Βρίσκει την τοπική IP του raspberry
def get_local_ip():
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        s.connect(('192.255.255.255', 1))
        IP = s.getsockname()[0]
    except:
        IP = '127.0.0.1'
    finally:
        s.close()
    return IP

local_ip = get_local_ip() #Βρες τοπική IP
hostName = local_ip #Βάλε το hostname για http server
serverPort = 8080   #Πόρτα για webserver

#Δημιουργία στιγμιοτύπου
webServer = HTTPServer((hostName, serverPort), MyServer)
print("Server started http://%s:%s" % (hostName, serverPort))

#Συνάρτηση για να ξεκινήσει ο Web Server. Θα ξεκινήσει σε άλλο Thread
def create_httpserver():
    webServer.serve_forever()

GPIO.setmode(GPIO.BCM) #Τρόπος αρίθμησης
GPIO.setwarnings(False) #Να μην εμφανίζει προειδοποιήσεις
GPIO.setup(RED_LED, GPIO.OUT) #Να γίνει έξοδος
GPIO.output(RED_LED, GPIO.LOW) #Αρχικά να είναι σβηστό

#Ξεκινάει τον WebServer σε thread
t = Thread(target = create_httpserver)
t.start()

#Για πάντα
try:
    while(True):
        time.sleep(1) #Καθυστέρησε μισό δευτερόλεπτο
#Αν πατηθεί Ctrl + C να σταματήσει
except KeyboardInterrupt:
    webServer.server_close()
    print("Server stopped.")
```

Ακολουθεί ο κώδικας σε μορφή κειμένου για εύκολη αντιγραφή.

```
#nmap -sP 192.168.42.0/24 στο Linux να βρω την IP του Raspberry
from http.server import BaseHTTPRequestHandler, HTTPServer
import time, socket
import RPi.GPIO as GPIO
from threading import Thread

RED_LED = 18 #Καλύτερα να χρησιμοποιούμε ονόματα αντί αριθμών

#Η σελίδα που θα προβληθεί σε HTML και λίγο Java Script
#Όπου υπάρχει %s θα αντικατασταθεί με τις παραμέτρους κατά την αποστολή της σελίδας
p1='''<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
 <title>Δοκιμή με Web Server</title>
 <script type="text/javascript">
   function set_RED(s)
     {
      if (s == 1)
          location.href = "/ctrl:RED=1";
      else if (s == 0)
          location.href = "/ctrl:RED=0";
     }
 </script>
</head>
<body>
 <H2>Έλεγχος εξόδου μέσω Browser</H2>
 <button type="button" id="RED_ON" onclick="set_RED(1)">Άναψε</button>  
 <button type="button" id="RED_OFF" onclick="set_RED(0)">Σβήσε</button>  <br><br>
 Το <span style="background-color:red; padding:3px;"> κόκκινο LED </span> είναι %s
</body>
</html>'''

state = "Σβηστό" #Αρχικά η κατάσταση του LED είναι σβηστό

#Ορισμός κλάσης του Web Server
class MyServer(BaseHTTPRequestHandler):
    def do_GET(self): #Αν υπάρχει αίτημα GET
        global state
        #Αν ζητήται η αρχική σελίδα χωρίς παραμέτρους
        if self.path == '/':
            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()
            self.wfile.write(bytes(p1 %(state), "utf-8")) #Στείλε και αντικατέστησε την μία παράμετρο με την μεταβλητή state
        #Αν ζητήται με την διεύθυνση /ctrl
        elif '/ctrl' in self.path:
            cmd = self.path.split(":") #Χώρισε το path βάσει της ':'
            #Αν υπάρχει το get /ctrl:RED=1 θα ανάψει
            if "RED=1" in cmd[1]:
                state = '<span style="background-color:yellow; padding:3px;"> Αναμμένο </span>'
                GPIO.output(RED_LED, GPIO.HIGH) #Άναψε το LED
                self.send_response(200)
                self.send_header("Content-type", "text/html")
                self.end_headers()
                self.wfile.write(bytes(p1 %(state), "utf-8"))
            #Αν υπάρχει το get /ctrl:RED=0 θα σβήσει
            if "RED=0" in cmd[1]:
                state = '<span style="background-color:black; color:white; padding:3px;"> Σβηστό </span>'
                GPIO.output(RED_LED, GPIO.LOW) #Σβήσε το LED
                self.send_response(200)
                self.send_header("Content-type", "text/html")
                self.end_headers()
                self.wfile.write(bytes(p1 %(state), "utf-8"))

#Βρίσκει την τοπική IP του raspberry
def get_local_ip():
         s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
         try:
                  s.connect(('192.255.255.255', 1))
                  IP = s.getsockname()[0]
         except:
                  IP = '127.0.0.1'
         finally:
                  s.close()
         return IP

local_ip = get_local_ip() #Βρες τοπική IP
hostName = local_ip #Βάλε το hostname για http server
serverPort = 8080   #Πόρτα για webserver

#Δημιουργία στιγμιοτύπου
webServer = HTTPServer((hostName, serverPort), MyServer)
print("Server started http://%s:%s" % (hostName, serverPort))

#Συνάρτηση για να ξεκινήσει ο Web Server. Θα ξεκινήσει σε άλλο Thread
def create_httpserver():
         webServer.serve_forever()

GPIO.setmode(GPIO.BCM) #Τρόπος αρίθμησης
GPIO.setwarnings(False) #Να μην εμφανίζει προειδοποιήσεις
GPIO.setup(RED_LED, GPIO.OUT) #Να γίνει έξοδος
GPIO.output(RED_LED, GPIO.LOW) #Αρχικά να είναι σβηστό

#Ξεκινάει τον WebServer σε thread
t = Thread(target = create_httpserver)
t.start()

#Για πάντα
try:
    while(True):
        time.sleep(1) #Καθυστέρησε μισό δευτερόλεπτο
#Αν πατηθεί Ctrl + C να σταματήσει
except KeyboardInterrupt:
         webServer.server_close()
         print("Server stopped.")
```