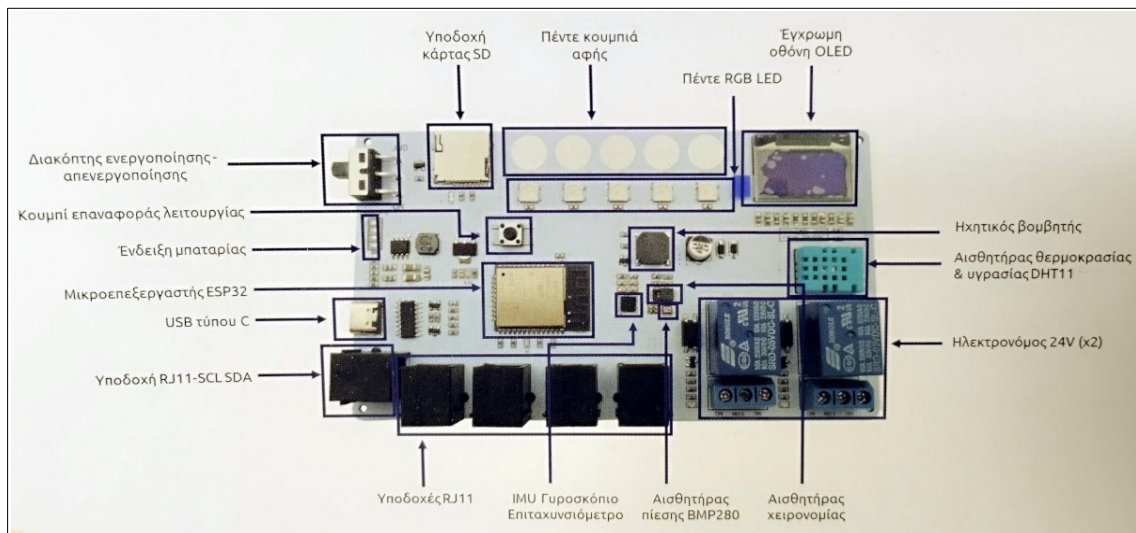


# Προγραμματισμός Micropython

στο σύστημα ARD:Icon II της Polytech



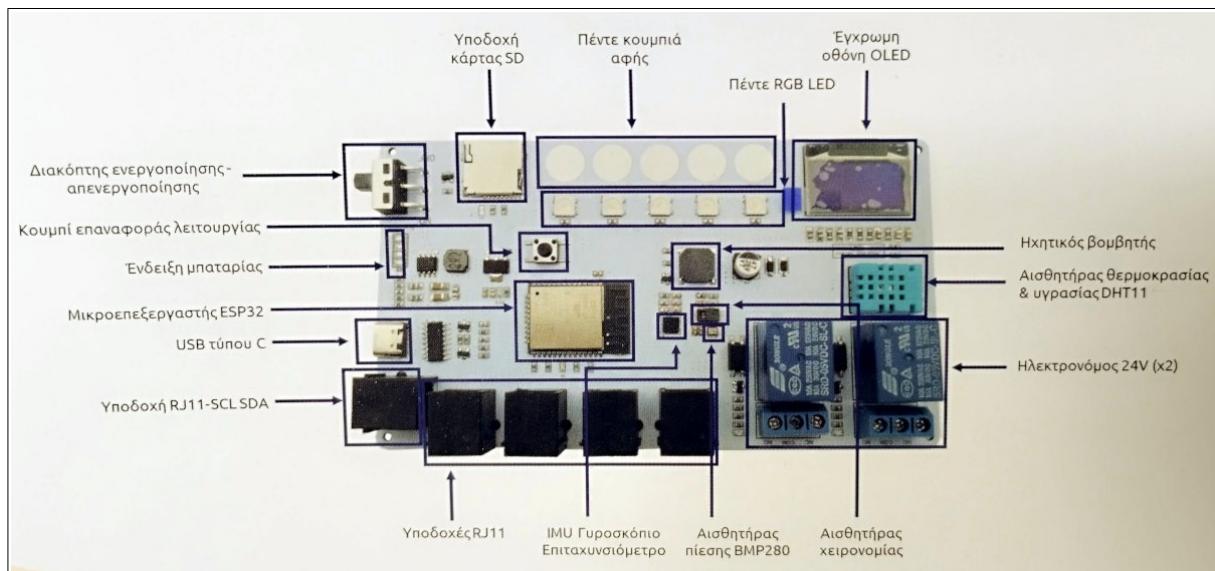
Έκδοση 1.0

**Σταύρος Σ. Φώτογλου ΠΕ86**  
Ε.Κ. Πρέβεζας – 1ο ΕΠΑ.Λ. Πρέβεζας

Πρέβεζα 2025

## MicroPython στο ARD:Icon II

Η πλακέτα **ARD:Icon II** υπάρχει στο κιτ ρομποτικής **S4T**. Έχει πολλές δυνατότητες και εγκατεστημένους πολλούς αισθητήρες. Με μια μικρή τροποποίηση των μπλε καλωδίων με τους συνδετήρες RJ12 (παραλλαγή με το latch στην άκρη) μπορούμε να χρησιμοποιήσουμε σχεδόν το σύνολο των αισθητήρων του κιτ. Στην πλακέτα μπορούμε να εγκαταστήσουμε την Micropython για πιο εύκολο προγραμματισμό ειδικά από τους μαθητές των ΕΠΑ.Λ. οι οποίοι διδάσκονται την Python σύμφωνα με το αναλυτικό πρόγραμμα.



### Τρόπος εγκατάστασης της Micropython

#### α. Ubuntu Linux 24.04 LTS

Αρχικά φτιάχνουμε ένα εικονικό περιβάλλον της Python 3.12.x από το τερματικό με `python3 -m venv ~/micropython`. Χρησιμοποιούμε το περιβάλλον με `source micropython/bin/activate`. Το prompt τώρα έγινε: `(micropython) stavros@stavLinux:~$`. Ελέγχω το περιβάλλον με `which python` και απαντά:

`/home/stavros/micropython/bin/python.`

Εγκατάσταση του esptool με `pip3 install esptool`.

Εντοπισμός σειριακής σύνδεσης με την εντολή `ls -l /dev/serial/by-path`. Εμφανίζει συνήθως `ttUSB0` ή `ttUSB1`.

#### Έλεγχος επικοινωνίας

Δίνω την εντολή `python -m esptool --port /dev/ttUSB0 flash-id` και απαντάει:

```
esptool v5.0.1
Connected to ESP32 on /dev/ttUSB0:
Chip type:      ESP32-D0WDQ6 (revision v1.1)
Features:       Wi-Fi, BT, Dual Core + LP Core, 240MHz, Vref calibration in eFuse, Coding Scheme None
Crystal frequency: 40MHz
MAC:           ec:64:c9:a4:1b:10

Stub flasher running.

Flash Memory Information:
=====
Manufacturer:  5e
Device:         4016
Detected flash size: 4MB
Flash voltage set by a strapping pin: 3.3V

Hard resetting via RTS pin...
```

#### Σβήσιμο της flash

Για να σβήσουμε ότι υπάρχει στην μνήμη flash γράφουμε `python -m esptool --port /dev/ttUSB0 erase-flash` και επιστρέφει:

```
esptool v5.0.1
Connected to ESP32 on /dev/ttyUSB0:
Chip type:      ESP32-D0WDQ6 (revision v1.1)
Features:       Wi-Fi, BT, Dual Core + LP Core, 240MHz, Vref calibration in eFuse, Coding Scheme None
Crystal frequency: 40MHz
MAC:            ec:64:c9:a4:1b:10

Stub flasher running.

Flash memory erased successfully in 12.4 seconds.

Hard resetting via RTS pin...
```

**Κατεβάζουμε** την τελευταία έκδοση της Micropython από εδώ: [https://micropython.org/download/ESP32\\_GENERIC/](https://micropython.org/download/ESP32_GENERIC/) . Την στιγμή που γράφεται ο οδηγός η τρέχουσα έκδοση είναι η 1.25.0. Σε μερικά παραδείγματα ίσως να φαίνεται η έκδοση 1.24.1 αλλά δεν υπάρχει κανένα πρόβλημα όποια έκδοση και να εγκαταστήσετε.

### Εγκατάσταση

Για να γράψουμε την micropython στην μνήμη Flash του ESP32 γράφουμε:

```
python3 -m esptool --port /dev/ttyUSB0 --baud 460800 write-flash -z 0x1000
~/Διήσεις/ESP32_GENERIC-20250415-v1.25.0.bin και επιστρέφει:
```

```
esptool v5.0.1
Connected to ESP32 on /dev/ttyUSB0:
Chip type:      ESP32-D0WDQ6 (revision v1.1)
Features:       Wi-Fi, BT, Dual Core + LP Core, 240MHz, Vref calibration in eFuse, Coding Scheme None
Crystal frequency: 40MHz
MAC:            ec:64:c9:a4:1b:10

Stub flasher running.
Changing baud rate to 460800...
Changed.

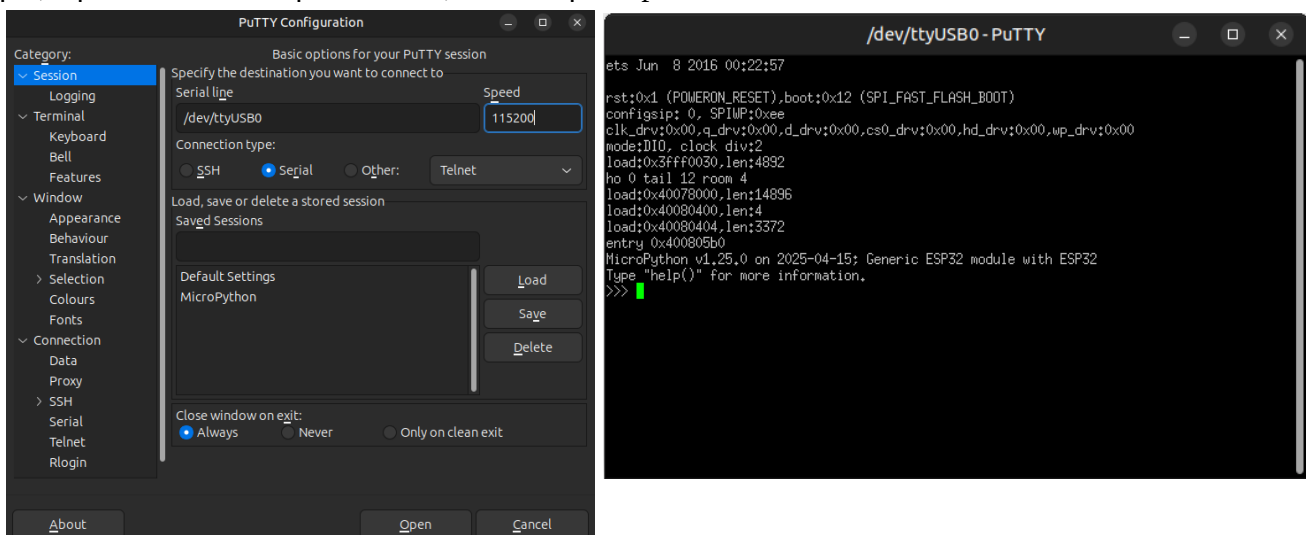
Configuring flash size...
Flash will be erased from 0x00001000 to 0x001a0fff...
Wrote 1702240 bytes (1117021 compressed) at 0x00001000 in 28.6 seconds (475.9 kbit/s) .
Hash of data verified.

Hard resetting via RTS pin...
```

Πατάμε **exit** για να βγούμε από το venv και μετά κλείνουμε το τερματικό.

### Δοκιμή

Ανοίγουμε το putty και επιλέγουμε Serial μετά στο Serial line βάζουμε /dev/ttyUSB0 και στο speed βάζουμε 115200. Πατάμε κάτω δεξιά το κουμπί Open.



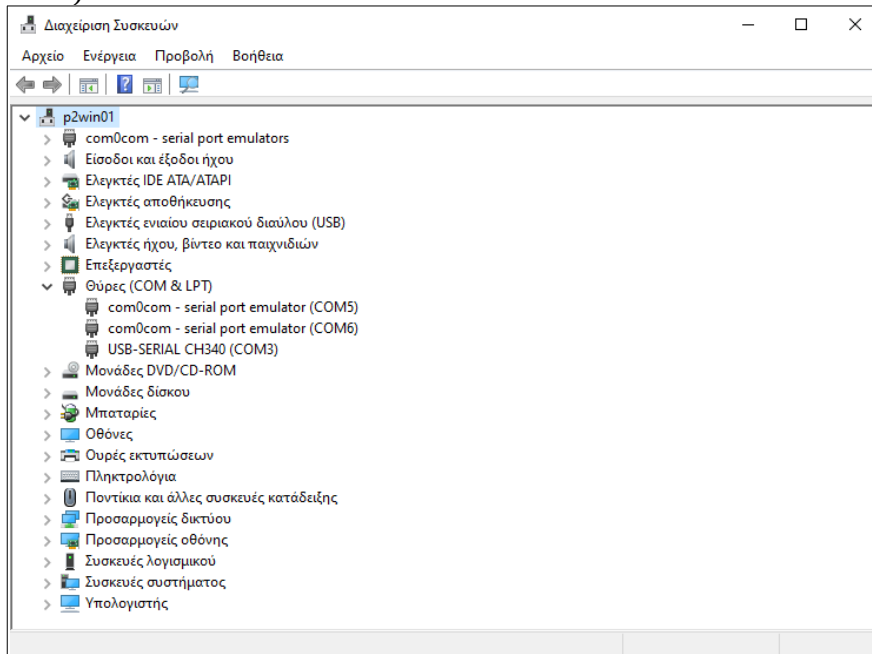
Πατάμε το button reset στην πλακέτα και αν εμφανίσει ότι στην εικόνα πάνω δεξιά σημαίνει ότι η Micropython εγκαταστάθηκε επιτυχώς.

## β. Windows

Στην αναζήτηση γράφουμε cmd και εκτελούμε την γραμμή εντολών.

Εκεί γράφουμε `pip install esptool`.

Από την διαχείριση συσκευών εντοπίζουμε την σειριακή θύρα της πλακέτας ανοίγοντας τον κλάδο Θύρες (COM & LPT).

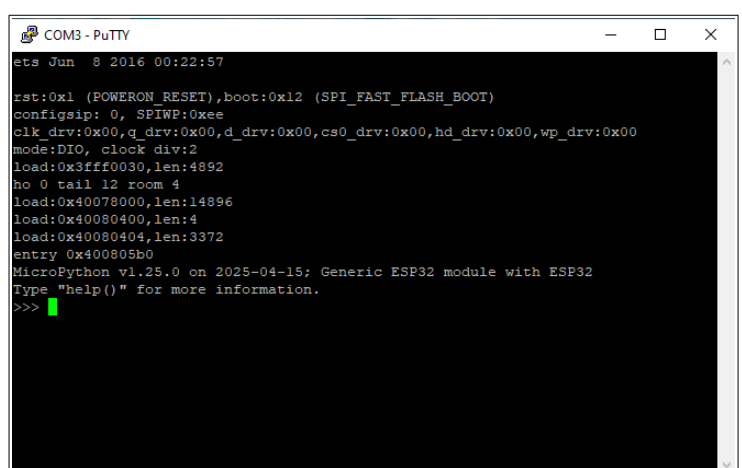
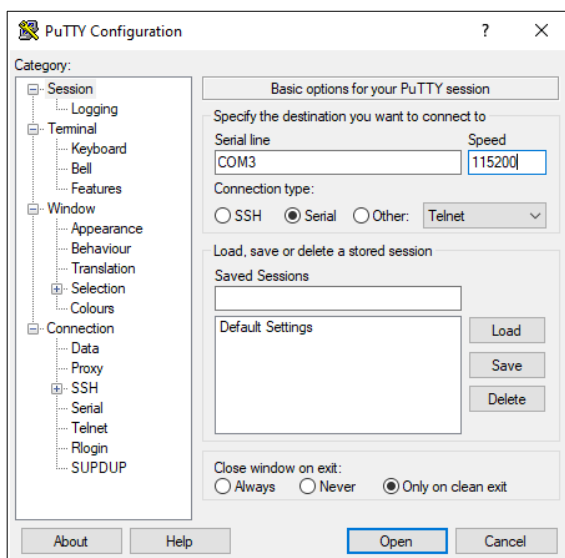


Στην περίπτωση μας είναι η COM3.

Από εδώ και μετά ακολουθούμε την ίδια διαδικασία με το linux, απλώς μόνο αλλάζουμε το όνομα της θύρας σε com3. Για παράδειγμα για να δούμε πληροφορίες του ESP32 module γράφουμε:

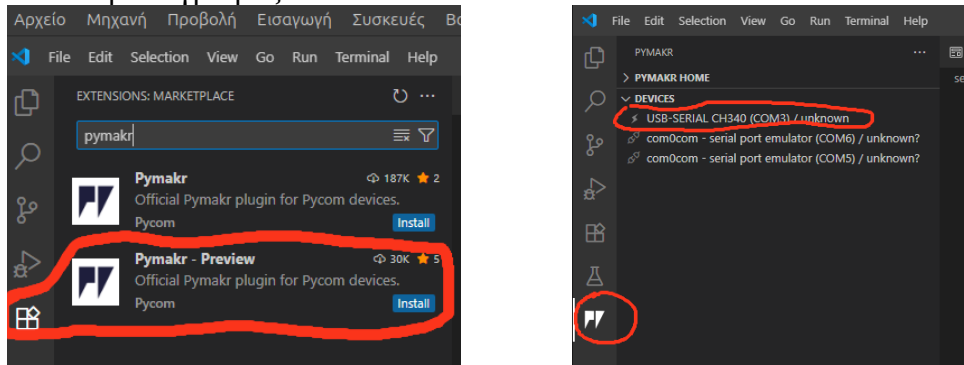
**`python -m esptool --port com3 flash-id`**

Αφού ακολουθήσουμε την παραπάνω διαδικασία ανοίγουμε το putty και ελέγχουμε αν όλα πήγαν καλά.

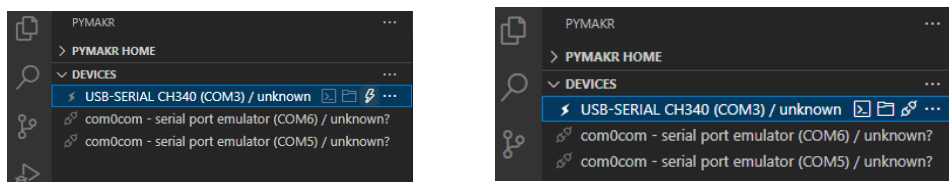


## Χρήση του IDE VS Code με την Micropython

Στο vscode πάμε στα extensions και κάνουμε αναζήτηση για το pymakr το οποίο εγκαθιστούμε. Προσοχή στα windows λειτουργεί η έκδοση Pymakr – Preview 2.25.2 με ημερομηνία τελευταίας έκδοσης 7-9-22. Μόλις τελειώσει η εγκατάσταση εμφανίζει αριστερά ένα νέο εργαλείο σαν διπλά εισαγωγικά. Πατάμε το εργαλείο και μας εμφανίζει τις σειριακές θύρες που είναι συνδεδεμένο το ESP32. Στο παράδειγμά μας COM3.

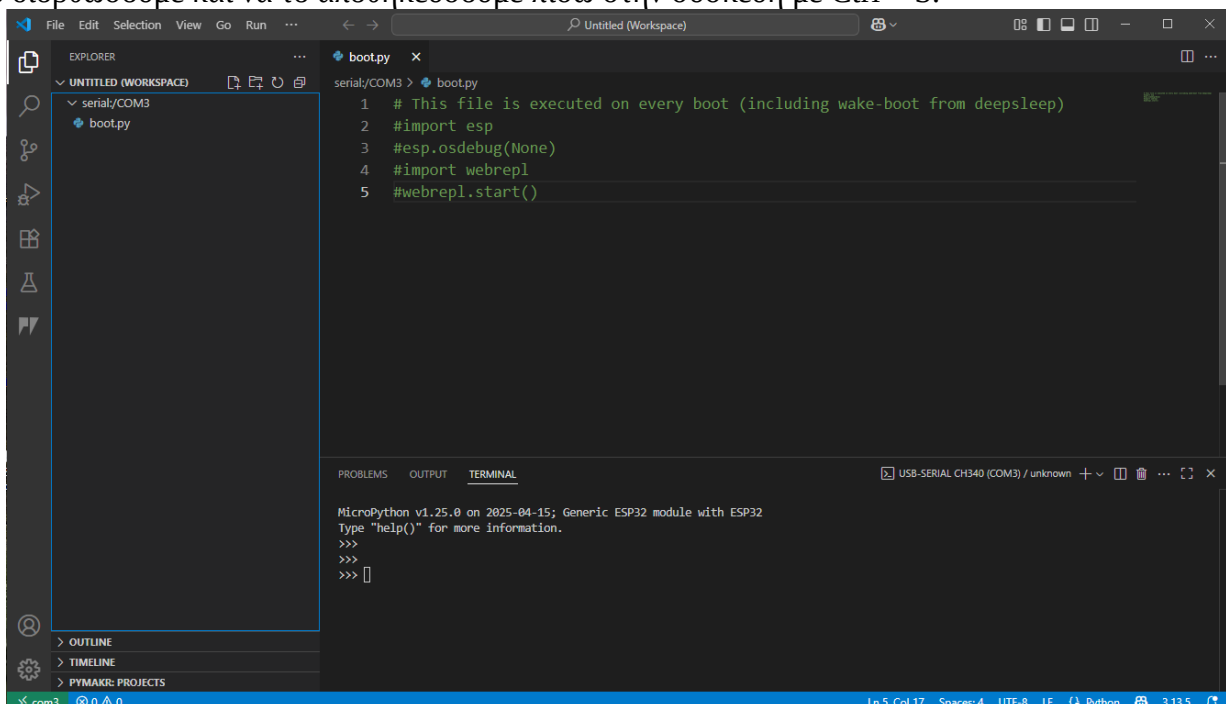


Επιλέγουμε την COM3 και πατάμε το σύμβολο του κεραυνού (connect device). Τότε ενεργοποιούνται τα εικονίδια του τερματικού του διαχειριστή αρχείων και της αποσύνδεσης.



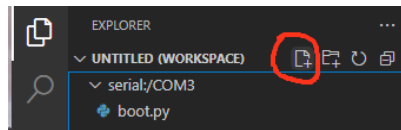
Αν πατήσουμε το τερματικό ανοίγει κάτω στο κέντρο το τερματικό της micropython και αν πατήσουμε τον διαχειριστή αρχείων ανοίγει τα περιεχόμενα της συσκευής στον διαχειριστή αρχείων.

Αν πατήσουμε το εργαλείο δεξιά πάνω που είναι ο Explorer, βλέπουμε τα αρχεία στο σύστημα αρχείων της συσκευής που αρχικά είναι μόνο το boot.py. Αν πατήσουμε πάνω σ' αυτό μπορούμε να το διορθώσουμε και να το αποθηκεύσουμε πίσω στην συσκευή με Ctrl + S.



## Ο κύκλος ανάπτυξης

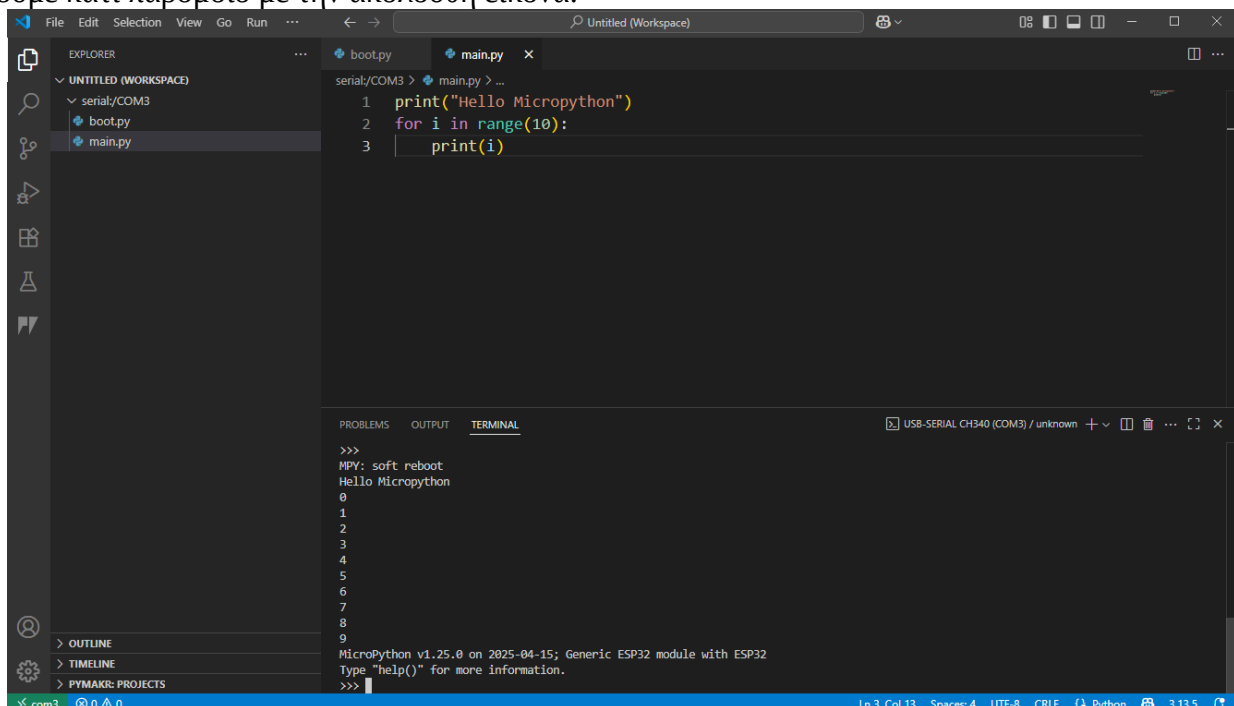
Πατάμε το εργαλείο με την σελίδα και το '+' ώστε να φτιάξουμε νέο αρχείο. Εκεί ζητάει όνομα και γράφουμε main.py. Τώρα στο κεντρικό τμήμα της οθόνης ανοίγει νέο tab στο οποίο μπορούμε να γράψουμε κώδικα σε python.



Γράφουμε για δοκιμή το παρακάτω πρόγραμμα και πατάμε Ctrl + S ώστε να αποθηκευτεί στην συσκευή.

```
print("Hello Micropython")
for i in range(10):
    print(i)
```

Στη συνέχεια κάνουμε κλικ στο κάτω μέρος που είναι το τερματικό και πατάμε Ctrl + D ώστε να εκτελεστεί ο κώδικας στην συσκευή. Αν δεν υπάρχουν συντακτικά λάθη θα εκτελεστεί και θα δούμε κάτι παρόμοιο με την ακόλουθη εικόνα:



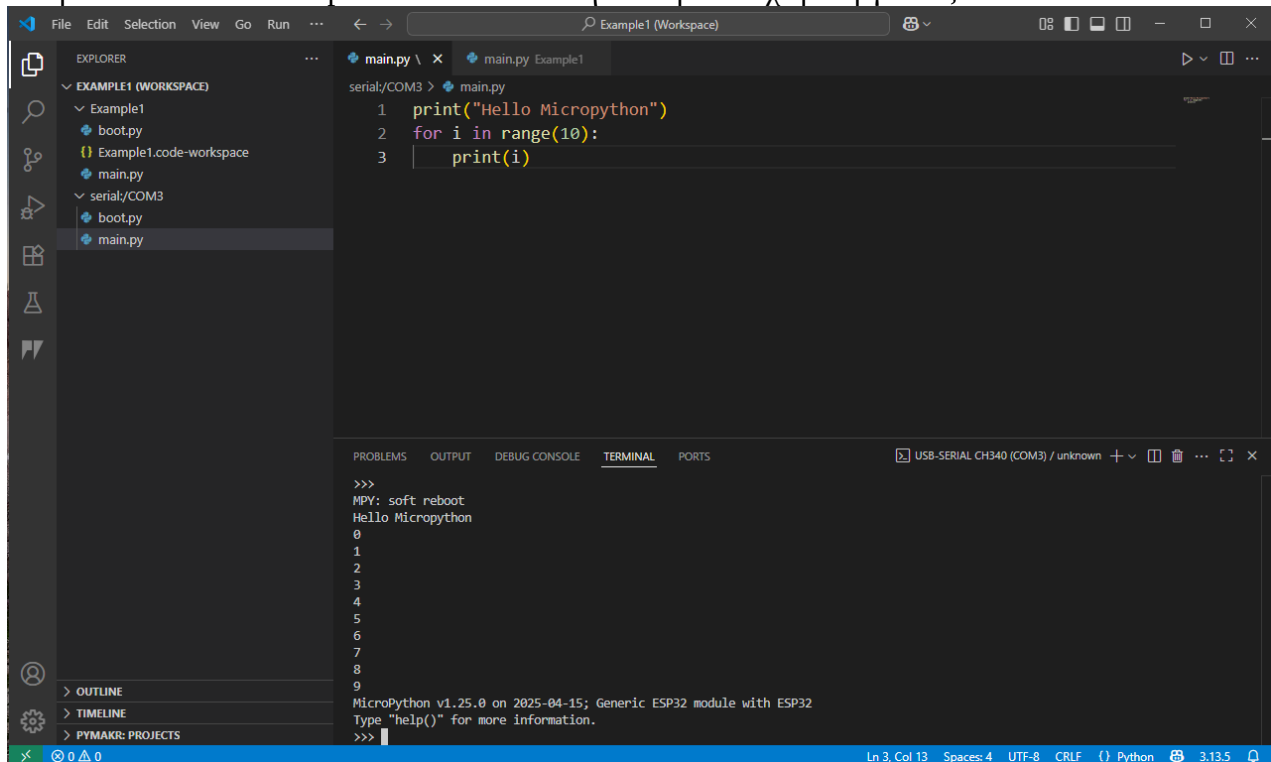
Αν θέλουμε να διορθώσουμε κάνουμε κλικ στο τμήμα του κώδικα, αποθηκεύουμε τις αλλαγές με CTRL + S, κάνουμε κλικ στο κάτω τμήμα που είναι το τερματικό, εκτελούμε με CTRL + D κ.ο.κ. Αν το πρόγραμμα σκόπιμα ή από λογικό λάθος μπαίνει σε ατέρμονα βρόχο μπορούμε να σταματήσουμε την εκτέλεση με CTRL + C. Προσοχή αν το πρόγραμμα βρίσκεται σε ατέρμονα βρόχο δεν μπορεί να αποθηκεύσει αλλαγές που γίνονται στον κώδικα. Δηλαδή για να αποθηκεύσουμε αλλαγές στην συσκευή το πρόγραμμα πρέπει να είναι σταματημένο.

## Τρόπος προγραμματισμού των δραστηριοτήτων

Αρχικά στον κατάλογο του χρήστη δημιουργούμε φάκελο με όνομα Micropython. Μέσα σ' αυτόν νέο φάκελο με όνομα π.χ. Example1. Από το vscode πάμε στον Explorer ή File – Open folder και επιλέγουμε τον φάκελο Example1. Από το pycharm ανοίγουμε την σύνδεση όπως πριν και πατάμε το εικονίδιο του file manager και του τερματικού. Τώρα στον Explorer έχουμε δύο τοποθεσίες, την τοπική του χρήστη Example1 και την απομακρυσμένη π.χ. COM3. Τώρα πατώντας δεξί κλικ σε οποιοδήποτε αρχείο και μετά pycharm μπορούμε να αντιγράψουμε αρχεία μεταξύ των δύο περιοχών. Αυτό είναι χρήσιμο αν έχουμε βιβλιοθήκες και πρέπει να τις αντιγράψουμε στο σύστημα. Επίσης



μπορούμε να έχουμε ένα αντίγραφο ασφαλείας των αρχείων τοπικά στον υπολογιστή μας. Τέλος πατάμε File – Save workspace as ... και αποθηκεύουμε τον χώρο εργασίας τοπικά.



## Το περιβάλλον της Micropython

Με το πρόγραμμα τερματικού putty ή μέσα από το extension **PyMakr** του vscode ανοίγουμε ένα τερματικό. Η ταχύτητα επικοινωνίας είναι 115200bps.

Με την εντολή help() βλέπουμε οδηγίες για την χρήση του shell της Micropython.

Βασικά εμάς μας ενδιαφέρουν οι συντομεύσεις **CTRL-C** για να διακόπτουμε την εκτέλεση ενός προγράμματος και **CTRL-D** για να κάνουμε reset την πλακέτα.

Με την εντολή **help('modules')** βλέπουμε τα εγκατεστημένα modules της Micropython.

```
MicroPython v1.24.1 on 2024-11-29; Generic ESP32 module with ESP32
Type "help()" for more information.
>>>
>>> help('modules')
_main_          bluetooth      heapq          select
_asyncio        btree          inisetup       socket
_boot           builtins       io             ssl
_espnow         collections    json           struct
_onewire        cryptolib      math           sys
_thread         deflate        micropython    time
_webrepl        dht            mip/__init__   tls
aioespnow       ds18x20        neopixel       uasyncio
apal06          errno          network        ctypes
array           esp            ntptime        umqtt/robust
asyncio/__init__ esp32           onewire        umqtt/simple
asyncio/core    espnow         os             upysh
asyncio/event   flashbdev     platform       urequests
asyncio/funcs   framebuffer    random         vfs
asyncio/lock    gc             re             webrepl
asyncio/stream  hashlib        requests/__init__ webrepl_set
binascii        Plus any modules on the filesystem
websocket
```

Αν θέλουμε να χρησιμοποιήσουμε ένα module γράφουμε **import <όνομα>** π.χ.

```
import machine
```

Αν γράψουμε **dir(machine)** θα εμφανιστούν όλες οι κλάσεις του module.

```
>>> dir(machine)
['_class_', 'name', 'ADC', 'ADCBLOCK', 'DAC', 'DEEPSLEEP', 'DEEPSLEEP_RESET', 'EXT0_WAKE', 'EXT1_WAKE',
'HARD_RESET', 'I2C', 'I2S', 'PIN_WAKE', 'PWM', 'PWRON_RESET', 'Pin', 'RTC', 'SDCard', 'SLEEP', 'SOFT_RESET',
'SPI', 'Signal', 'SoftI2C', 'SoftSPI', 'TIMER_WAKE', 'TouchPad', 'UART', 'ULP_WAKE',
'WDT', 'WDT_RESET', '__dict__', 'bitstream', 'deepsleep', 'dht_readinto', 'disable_irq', 'enable_irq', 'freq',
'idle', 'lightsleep', 'mem16', 'mem32', 'mem8', 'reset', 'reset_cause', 'sleep', 'soft_reset', 'time_pulse_us',
'unique_id', 'wake_reason']
```

Αν επιλέξουμε μια κλάση π.χ. την Pin και γράψουμε `dir(machine.Pin)` θα δούμε όλες τις μεθόδους της κλάσης.

```
>>> dir(machine.Pin)
['_class_', '__name__', 'value', 'DRIVE_0', 'DRIVE_1', 'DRIVE_2', 'DRIVE_3', 'IN', 'IRQ_FALLING', 'IRQ_RISING', 'OPEN_DRAIN', 'OUT', 'PULL_DOWN', 'PULL_UP', 'WAKE_HIGH', 'WAKE_LOW', '__bases__', '__dict__', 'board', 'init', 'irq', 'off', 'on']
```

### Έλεγχος ελεύθερου χώρου στο File system

Κάνουμε εισαγωγή το module os με `import os` και μετά καλούμε την μέθοδο `os.statvfs('/')`. Αυτή επιστρέφει μια πλειάδα με το πρώτο στοιχείο να είναι το μέγεθος του block, το τρίτο είναι ο συνολικός χώρος και το τέταρτο ο ελεύθερος χώρος.

```
>>> import os
>>>
>>> os.statvfs('/')
(4096, 4096, 512, 483, 483, 0, 0, 0, 255)
```

Δηλαδή ο συνολικός χώρος είναι  $4096 \times 512 = 2097152 = 2 \text{ MB}$  και ο ελεύθερος χώρος είναι  $4096 \times 483 = 1978368 \text{ bytes}$ .

Αν γράψω `stat = os.statvfs('/')` και μετά

`print((stat[2] - stat[3]) * stat[0])` εμφανίζει την δεσμευμένη μνήμη του συστήματος αρχείων (Flash).

```
>>> stat = os.statvfs('/')
>>> print((stat[2] - stat[3]) * stat[0])
118784
```

### Εμφάνιση αρχείων στο σύστημα αρχείων

Με `os.listdir()` βλέπουμε τα αρχεία που υπάρχουν στο σύστημα αρχείων την μνήμης flash.

```
>>> os.listdir()
['OLED-ESP32.code-workspace', 'bmp280.py', 'boot.py', 'courier20.py', 'font10.py', 'font6.py', 'freesans20.py', 'main.py', 'ssd1306.py', 'writer.py']
>>>
```

### Έλεγχος μνήμης RAM

Κάνουμε εισαγωγή το module gc (garbage collector) με `import gc` και μετά γράφουμε `gc.mem_free()`. Αν θέλουμε χειροκίνητα να κάνουμε εξοικονόμηση μνήμης γράφουμε `gc.collect()` και ελέγχουμε πάλι την ελεύθερη μνήμη. Παρακάτω βλέπουμε ότι έχουμε 142Kbytes ελεύθερης μνήμης. Τέλος η εντολή `gc.mem_alloc()` επιστρέφει την δεσμευμένη μνήμη Ram από τον κώδικα της Python.

```
>>> import gc
>>> gc.mem_free()
138944
>>> gc.collect()
>>> gc.mem_free()
142512
>>>
>>> gc.mem_alloc()
37552
```

### Συχνότητα λειτουργίας

Με την εντολή `print(machine.freq())` εμφανίζει την συχνότητα λειτουργίας της CPU που εξ' ορισμού είναι 160000000 (160MHz). Η μέγιστη είναι 240MHz και αλλάζει με την εντολή: `machine.freq(240000000)`

## Παράδειγμα 1

### Το πρώτο μας πρόγραμμα – blink

Θα γράψουμε ένα πολύ απλό πρόγραμμα που δεν είναι άλλο από το blink του Arduino αλλά σε γλώσσα Python. Φτιάχνουμε ένα νέο Project με όνομα Example1 και το αποθηκεύουμε σε κάποιο φάκελο. Επιλέγουμε συσκευή com της πλακέτας και γράφουμε στο αρχείο main.py τον παρακάτω κώδικα:

```
1 import time
2 from machine import Pin
3
4 Relay1 = Pin(16, Pin.OUT)
5 while True:
```



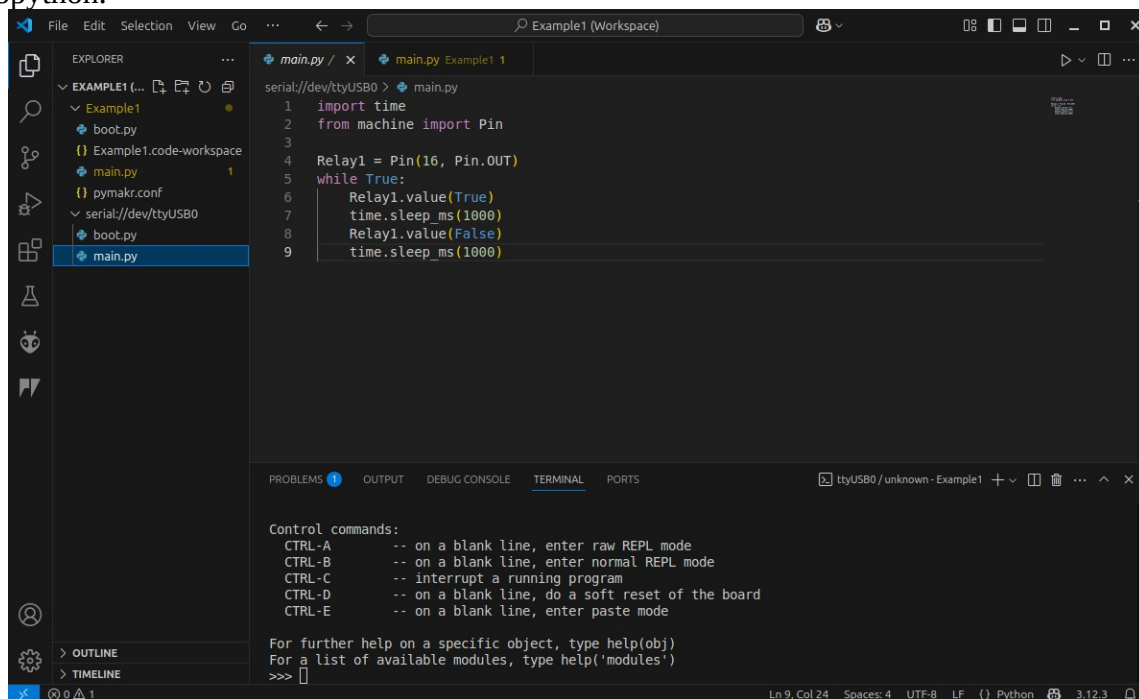
```

6 Relay1.value(True)
7 time.sleep_ms(1000)
8 Relay1.value(False)
9 time.sleep_ms(1000)

```

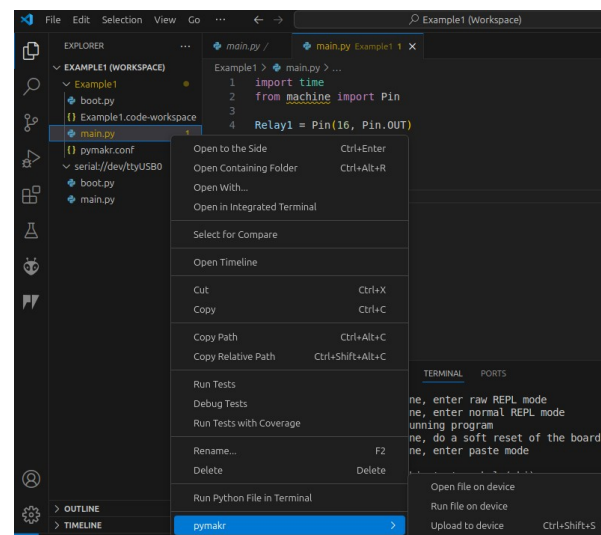
Το πρώτο Relay συνδέεται στον ακροδέκτη IO16. Στην 4η γραμμή γίνεται έξοδος και μέσα στην while η οποία εκτελείται για πάντα αρχικά το ενεργοποιεί, περιμένει 1sec, το απενεργοποιεί και περιμένει άλλο ένα sec.

Όπως βλέπουμε στον Explorer αριστερά έχουμε τον τοπικό φάκελο του έργου και το σύστημα αρχείων της συσκευής. Μπορούμε να διορθώνουμε το αρχείο είτε στον φάκελο του υπολογιστή είτε στην συσκευή απευθείας. Στον υπολογιστή φαίνονται και μερικά λάθη γιατί δεν γνωρίζει την ύπαρξη της βιβλιοθήκης machine. Πατάμε στην περιοχή τερματικού CTRL-D για να ξεκινήσει η εκτέλεση. Με CTRL-C σταματάμε την εκτέλεση. Για να λειτουργήσει ένα πρόγραμμα σε micropython είναι απαραίτητο το αρχείο **main.py** με τον κυρίως κώδικα, το **boot.py** το οποίο εκτελείται πρώτο κατά το reset και σ' αυτό συνήθως φορτώνουμε modules ή αρχικοποιούμε το υλικό. Στα παραδείγματά μας αυτό το αρχείο είναι κενό (μόνο σχόλια). Επίσης βάζουμε όσα αρχεία γίνονται import μέσα στον κώδικα του main και δεν υπάρχουν στα ενσωματωμένα modules της micropython.

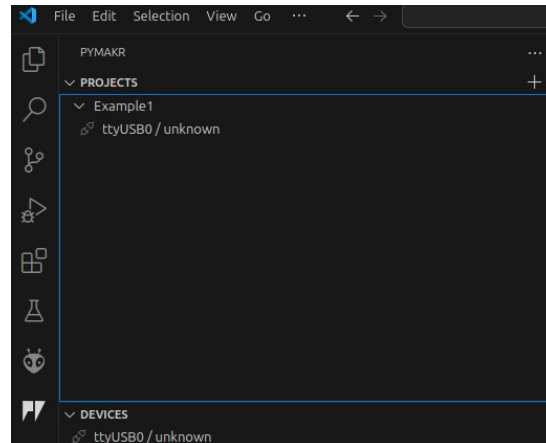


Αν κάνουμε αλλαγές στον υπολογιστή πρέπει να συγχρονίσουμε τα αρχεία πατώντας δεξί κλικ – pymakr – Upload to device.

Καλύτερα είναι να γράφουμε απευθείας στην συσκευή και στο τέλος να συγχρονίζουμε το φάκελο του έργου με αυτόν της συσκευής πατώντας το εικονίδιο του pymakr, κάνουμε focus την συσκευή (εδώ είναι Linux /dev/ttyUSB0) και πατάμε το τρίτο εικονίδιο Download project from device.



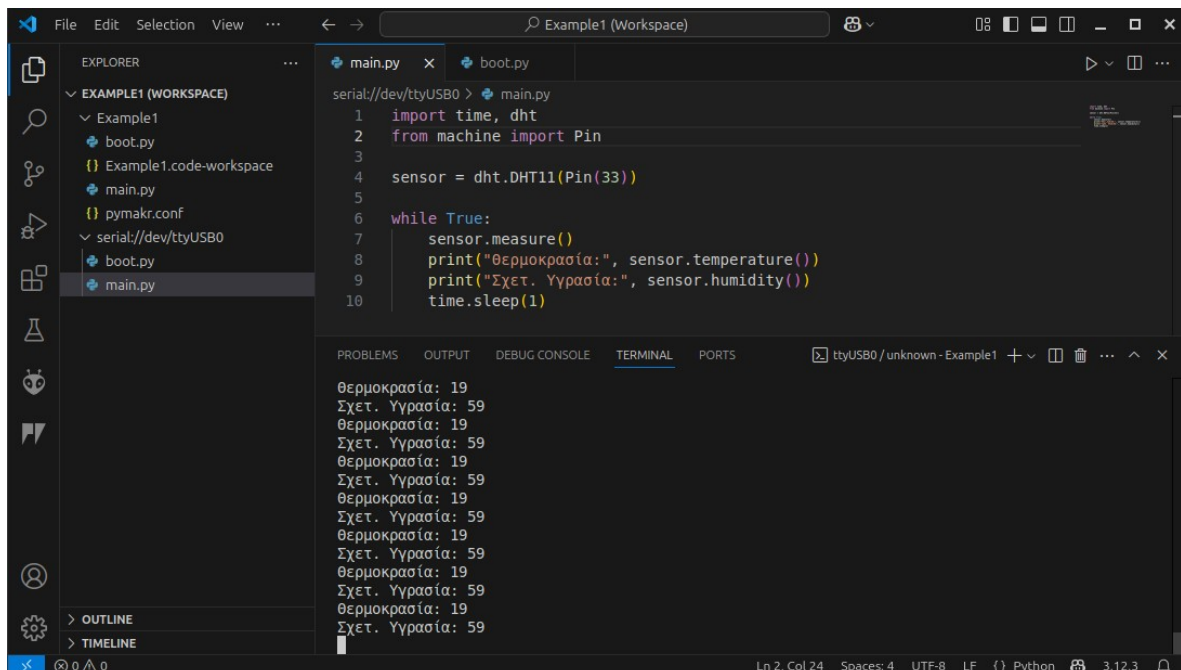
Προσοχή για να αποθηκευτούν οι αλλαγές στην συσκευή πρέπει να σταματάμε την εκτέλεση στο τερματικό με CTRL-C.



## Παράδειγμα 2 – Μέτρηση υγρασίας και θερμοκρασίας με το DHT11

Γράφουμε τον παρακάτω κώδικα στο main.py.

```
1 import time, dht
2 from machine import Pin
3
4 sensor = dht.DHT11(Pin(33))
5
6 while True:
7     sensor.measure()
8     print("Θερμοκρασία:", sensor.temperature())
9     print("Σχετ. Υγρασία:", sensor.humidity())
10    time.sleep(1)
```



## Παράδειγμα 3 – Έξοδος στα RGB Leds

Η πλακέτα διαθέτει 5 addressable RGB Leds με το chip WS2812B τα οποία συνδέονται στο pin IO25. Το module είναι το neopixel και είναι ήδη ενσωματωμένο μέσα στην micropython.

```
1 import machine, neopixel
2
3 np = neopixel.NeoPixel(machine.Pin(25), 5) #IO25, 5 x RGB Leds
4 #      R , G , B
```

```

5 np[0] = (255, 0, 0) #Κόκκινο 100%
6 np[1] = (125, 204, 223)
7 np[2] = (120, 153, 23)
8 np[3] = (255, 0, 153)
9 np[4] = (0, 0, 64) #Μπλε 25%
10 np.write()

```

### Τροποποίηση

Να τροποποιηθεί ο παραπάνω κώδικας ώστε να κάνει διάφορα οπτικά effects όπως fade in, fade out, Kit (υπό της της ασφάλτου) κλπ.

```

1 import machine, neopixel
2 import time
3
4 np = neopixel.NeoPixel(machine.Pin(25), 5) #IO25, 5 x RGB Leds
5
6 def blank():
7     np[0] = np[1] = np[2] = np[3] = np[4] = (0, 0, 0)
8     np.write()
9
10 def fade(R = 0, G = 0, B = 0):
11     #Fade in
12     for i in range(0, 256, 5):
13         r = i * R; g = i * G; b = i * B
14         for j in range(5):
15             np[j] = (r, g, b)
16             np.write()
17             time.sleep_ms(50)
18     #Fade out
19     for i in range(255, -1, -5):
20         r = i * R; g = i * G; b = i * B
21         for j in range(5):
22             np[j] = (r, g, b)
23             np.write()
24             time.sleep_ms(50)
25
26 def kit(color, times):
27     for k in range(times):
28         for i in range(5):
29             np[0] = np[1] = np[2] = np[3] = np[4] = (0, 0, 0)
30             np[i] = color
31             np.write()
32             time.sleep_ms(100)
33         for i in range(4, -1, -1):
34             np[0] = np[1] = np[2] = np[3] = np[4] = (0, 0, 0)
35             np[i] = color
36             np.write()
37             time.sleep_ms(100)
38
39 while(True):
40     fade(1, 0, 0)
41     fade(0, 1, 0)
42     fade(0, 0, 1)
43     kit((0, 255, 128), 5)
44     kit((255, 0, 128), 5)
45     kit((255, 0, 0), 5)
46     blank()
47     time.sleep(1)

```

### Παράδειγμα 4 – Ο βομβητής

Ο βομβητής (buzzer) συνδέεται στο pin IO5. Για να παραχθεί τόνος πρέπει να στείλουμε τετραγωνικό παλμό κάποιας ακουστικής συχνότητας στον βομβητή.

```

1 import time
2 from machine import Pin
3
4 beeper = Pin(5, Pin.OUT) #Ο βομβητής συνδέεται στο pin IO5
5
6 def beep():
7     for i in range(200):
8         beeper.value(1)
9         time.sleep_us(1000)
10        beeper.value(0)
11        time.sleep_us(1000)
12
13 beep()

```

Στην συνάρτηση beep() παράγουμε τόνο συχνότητας 500Hz για 200 κύκλους. Ο κάθε κύκλος διαρκεί 2msec (2000μsec), 1msec High και 1msec Low δηλαδή  $1 / (2 * 10^{-3}) = 500\text{Hz}$ . Εφόσον ο κάθε κύκλος διαρκεί 2msec και εμείς κάνουμε 200 επαναλήψεις τότε το beep διαρκεί  $2\text{msec} \times 200 = 400\text{msec}$  ή 0,4 δευτερόλεπτα.

### Τροποποίηση

Τροποποιήστε το παραπάνω πρόγραμμα ώστε στην συνάρτηση beep να δίνουμε παραμέτρους συχνότητας και διάρκειας και να παράγει τον αντίστοιχο τόνο.

```

1 import time
2 from machine import Pin
3
4 beeper = Pin(5, Pin.OUT) #Ο βομβητής συνδέεται στο pin IO5
5
6 def beep(freq = 500, dur = .5): #Default values
7     #Υπολογισμοί
8     period = 1.0 / freq #Περίοδος
9     half_per = int((period / 2) * 1000000) #Ημιπερίοδος
10    times = int(dur / period) #Αριθμός κύκλων
11    for i in range(times):
12        beeper.value(1)
13        time.sleep_us(half_per)
14        beeper.value(0)
15        time.sleep_us(half_per)
16
17 beep() #θα παραχθεί τόνος 500Hz για 0,5sec
18 beep(1000, .5) #1000Hz
19 beep(2000, .5) #2000Hz
20
21 for i in range(200, 4001, 10): #Σταδιακό ανέβασμα από 200 - 4000
22     beep(i, .003)
23 for i in range(4000, 195, -10): #Σταδιακό κατέβασμα από 4000 - 200
24     beep(i, .003)

```

## Παράδειγμα 5 – Ο αισθητήρας BMP280

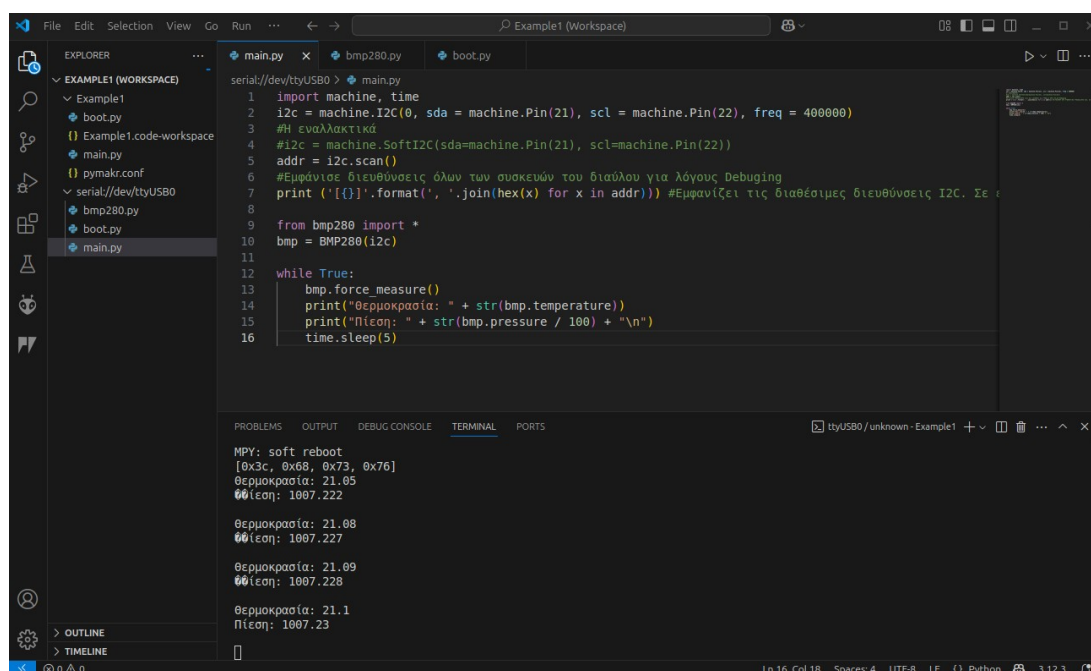
Ο αισθητήρας BMP280 της BOSCH είναι ενσωματωμένος στην πλακέτα και χρησιμοποιείται για μέτρηση ατμοσφαιρικής πίεσης και θερμοκρασίας. Συνδέεται με τον ESP32 μέσω του σειριακού διαύλου IIC ή I<sup>2</sup>C. Επειδή δεν είναι ενσωματωμένο το module στην Micropython θα πρέπει να το συμπεριλάβουμε στο σύστημα αρχείων ως ξεχωριστό αρχείο σε γλώσσα Python. Δηλαδή μέσα στο σύστημα αρχείων θα έχουμε το boot.py, το main.py και το bmp280.py το οποίο υπάρχει στο αρχείο zip στο Example5.

Η πλακέτα μας χρησιμοποιεί το pin IO21 για το σήμα SDA που είναι δικατευθυντήριο και το pin IO22 για το σήμα SCL που είναι το σήμα χρονισμού από το ESP32 (master) προς τις συσκευές slaves του διαύλου IIC. Το BMP280 έχει διεύθυνση 0x76.

```

1 import machine, time
2 i2c = machine.I2C(0, sda = machine.Pin(21), scl = machine.Pin(22), freq = 400000)
3 #Η εναλλακτικά
4 #i2c = machine.SoftI2C(sda=machine.Pin(21), scl=machine.Pin(22))
5 addr = i2c.scan()
6 #Εμφάνισε διευθύνσεις όλων των συσκευών του διαύλου για λόγους Debuging
7 print ('[{}]' .format(' '.join(hex(x) for x in addr))) #Εμφανίζει τις διαθέσιμες διευθύνσεις I2C. Σε εμάς είναι η 0x76
8
9 from bmp280 import *
10 bmp = BMP280(i2c)
11
12 while True:
13     bmp.force_measure()
14     print("Θερμοκρασία: " + str(bmp.temperature))
15     print("Πίεση: " + str(bmp.pressure / 100) + "\n")
16     time.sleep(5)

```



Πριν τις μετρήσεις μπορούμε να τροποποιήσουμε κάποιες παραμέτρους της βιβλιοθήκης όπως φαίνεται παρακάτω:

```
bmp.use_case(BMP280_CASE_INDOOR)
bmp.oversample(BMP280_OS_HIGH)

bmp.temp_os = BMP280_TEMP_OS_8
bmp.press_os = BMP280_PRES_OS_4

bmp.standby = BMP280_STANDBY_250
bmp.iir = BMP280_IIR_FILTER_2

bmp.spi3w = BMP280_SPI3W_ON
```

## Παράδειγμα 6 – Η μονόχρωμη οθόνη γραφικών OLED

Η πλακέτα διαθέτει οθόνη OLED 0,96” και ανάλυσης 128x64 pixels. Η οθόνη συνδέεται στον ίδιο διάυλο IIC με το BMP280 και έχει διεύθυνση 0x3c. Για να λειτουργήσει θα χρειαστούμε το module ssd1306.py το οποίο θα το βάλουμε μέσα στο σύστημα αρχείων. Το module βρίσκεται στο αρχείο zip στον φάκελο Example6.

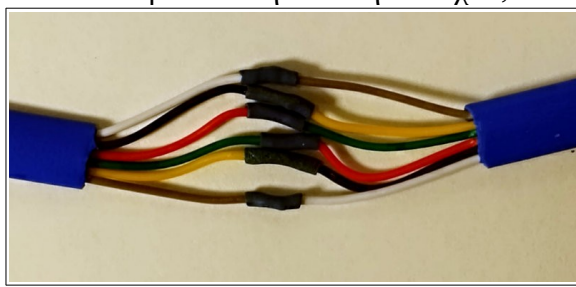
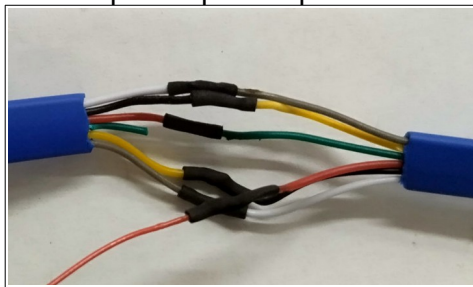
```
1 import machine, time
2 i2c = machine.I2C(0, sda = machine.Pin(21), scl = machine.Pin(22), freq = 400000)
3 #Η εναλλακτική
4 #i2c = machine.SoftI2C(sda=machine.Pin(21), scl=machine.Pin(22))
5 addr = i2c.scan()
6 #Εμφάνιση διεύθυνσης όλων των συσκευών του διαύλου για λόγους Debuging
7 print ('[{}]' .format(', '.join(hex(x) for x in addr))) #Εμφανίζει τις διαθέσιμες διεύθυνσεις I2C. Σε εμάς είναι η 0x76
8
9 from ssd1306 import SSD1306_I2C
10 #display = SSD1306_I2C(128, 64, i2c) #Δίνω μπορώ να βάλω διεύθυνση π.χ. SSD1306_I2C(128, 64, i2c, 0x3c)
11 display = SSD1306_I2C( width=128, height=64, i2c=i2c, addr=0x3c, external_vcc=False )
12 display.text('Hello', 5, 5) #θέση x, y
13 display.text('World', 5, 15, 1) #color 1/0
14 display.rect(0,0,127,63,1,0) #x, y, width, height, color=0/1, fill=0/1
15 display.show()
16
17 #display.fill(0) #Καθαρίζει οθόνη
18 #display.show()
```

## Παράδειγμα 7 – Η οθόνη χαρακτήρων 2x16

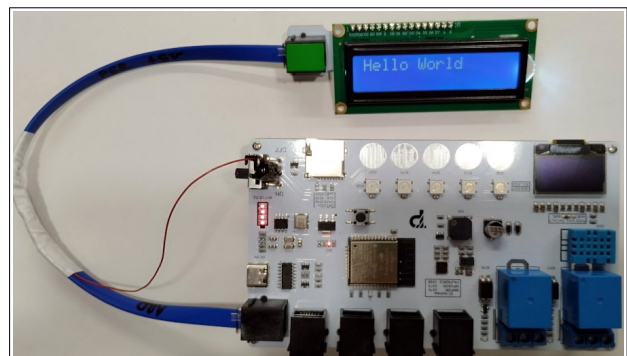
Μέσα στο κιτ υπάρχει και μία οθόνη χαρακτήρων 2 γραμμών και 16 στηλών. Τα περιφερειακά αυτά έχουν σχεδιαστεί για τον ελεγκτή arduino ARD:icon I και υπάρχει ένα πρόβλημα με τα καλώδια σύνδεσης τα οποία είναι αντεστραμμένα και την τάση λειτουργίας των περισσότερων η οποία είναι 5V ενώ η πλακέτα του ESP32 λειτουργεί στα 3,3V.

Το πρόβλημα λύνεται με δύο τρόπους:

**1. Μετατροπή καλωδίων RJ12.** Κόβουμε στην μέση ένα καλώδιο και αντιστρέψουμε τα χρώματα των εσωτερικών καλωδίων. Στην αριστερή εικόνα αριστερά είναι η πλακέτα ESP32 Icon II και δεξιά οποιοδήποτε περιφερειακό (αισθητήρας ή ενεργοποιητής) του Icon I. Στην αριστερή εικόνα δεν συνδέσαμε το πράσινο με το κόκκινο ώστε να δώσουμε 5V στην οθόνη και όχι 3,3V.

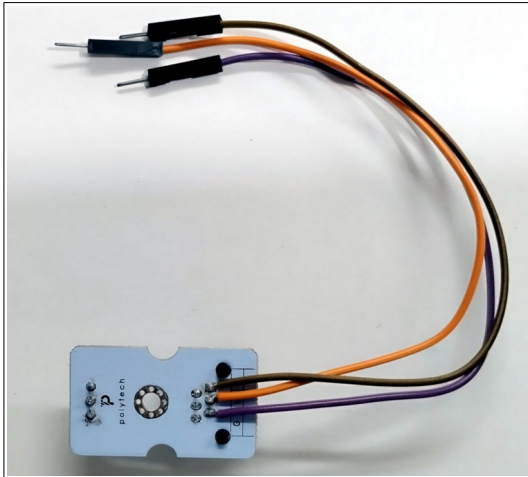


Στην διπλανή εικόνα φαίνεται η διασύνδεση της πλακέτας με την οθόνη χαρακτήρων 2x16. Η οθόνη λειτουργεί σωστά με τάση 5V την οποία την δίνουμε συνδέοντας ένα κροκοδειλάκι στην μεσαία επαφή του διακόπτη.

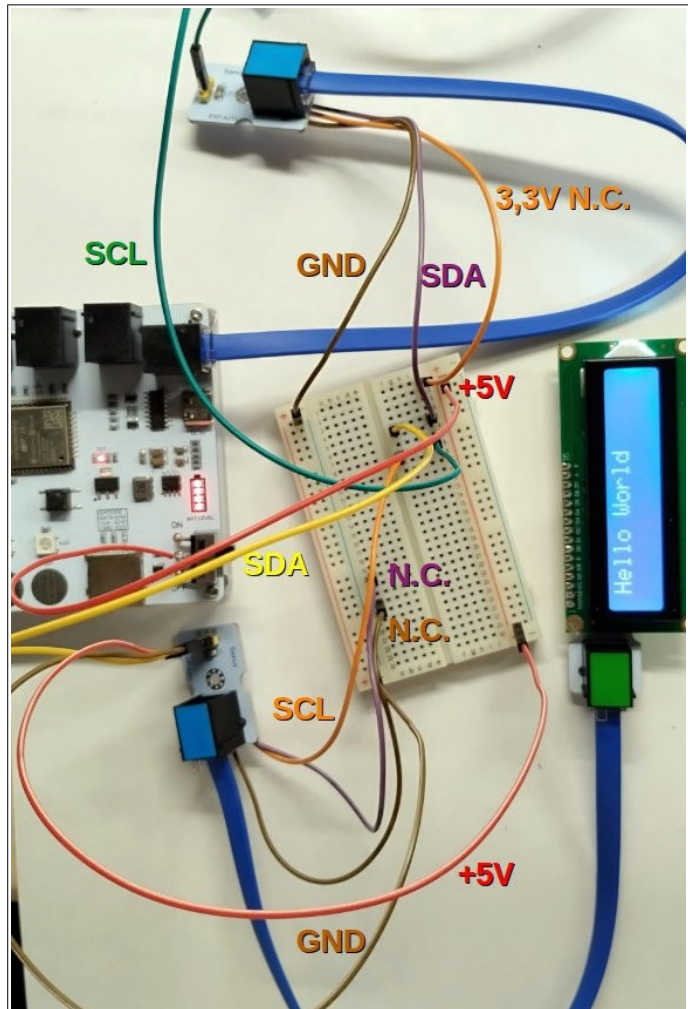




**2. Χρήση Breadboard χωρίς μετατροπή καλωδίων.** Θα χρειαστούμε δύο EXP-AJ11 (Προσαρμογέας RJ11) από τους τρεις που διαθέτει το kit. Επίσης θα χρειαστούμε τρία χρωματιστά καλώδια breadboard test (M/M) τα οποία τα κόβουμε στην μέση και τα κολλάμε στις EXP-AJ11 σύμφωνα με την παρακάτω εικόνα αριστερά. Για το icon II το πορτοκαλί είναι +3,3V, το καφέ **GND** και το μωβ είναι το **GPIO** αναλόγως με την θέση σύνδεσης. Στο παράδειγμά μας είναι το σήμα **SDA**. Ακόμη θα χρειαστούμε τέσσερα καλώδια dupont (F/M) όπως είναι (χωρίς να τα κόψουμε) και την breadboard.



Στην διπλανή εικόνα φαίνεται ο τρόπος διασύνδεσης των δύο τροποποιημένων EXP-AJ11, των καλωδίων RJ12, του icon II, της breadboard και της οθόνης χαρακτήρων. Προσοχή τα +5V τα παίρνουμε από την μεσαία επαφή του διακόπτη. Στο πάνω EXP-AJ11 το πράσινο καλώδιο (SCL) συνδέεται στο μεσαίο pin με ένδειξη V. Στο κάτω EXP-AJ11 το καφέ πάει στο G, το κόκκινο στο V και το κίτρινο (SDA) στο S. Τα καλώδια με σήμανση N.C. (Not Connected) δεν συνδέονται κάπου.



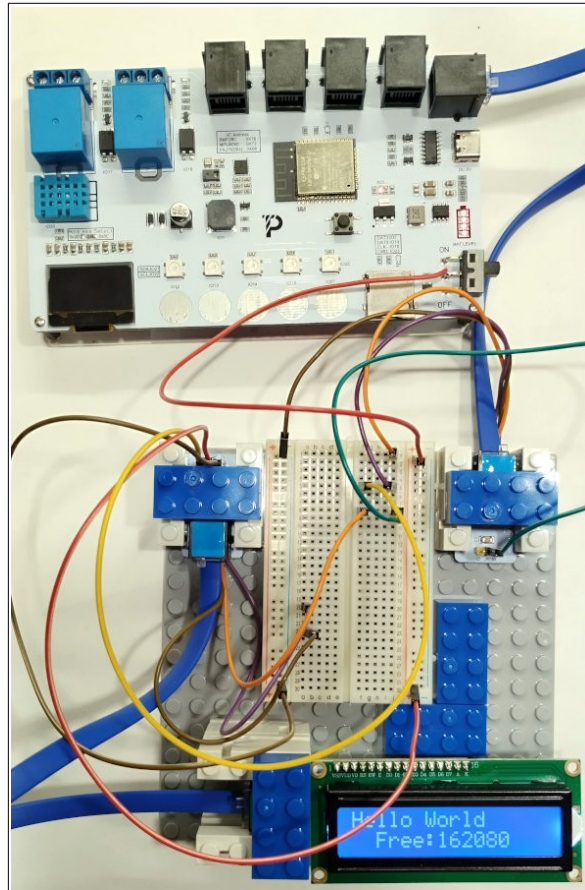
Για να λειτουργήσει πρέπει να συμπεριληφθεί το αρχείο lcd\_i2c8574.py μέσα στο σύστημα αρχείων.

```

1 # ----- Διεύθυνση I2C -----
2 I2C_Addr = 0x27 # Η διεύθυνση μπορεί να αλλάξει στο πίσω πλακετάκι του PCF8574
3
4 # ----- Διαστάσεις LCD οθόνης ----
5 LCD_Dim = (16, 2)
6
7 from machine import I2C, Pin
8 i2c = I2C(0, sda=Pin(21), scl=Pin(22), freq=100000)
9
10 import gc
11 from time import sleep
12 from lcd_i2c8574 import I2cLcd
13
14 lcd = I2cLcd(i2c, I2C_Addr, LCD_Dim)
15 gc.collect() #Ενθλογή σκουπιδιών
16 mfree1 = gc.mem_free() #Ελεύθερη μνήμη RAM
17
18 lcd.clear() #Καθαρισμός οθόνης και τοποθέτηση cursor στην θέση 0, 0
19 lcd.write('Hello World', end='')
20 lcd.move_to(2, 1) #Τοποθέτηση cursor στην 3η στήλη, 2η γραμμή
21 lcd.write("Free:" + str(mfree1), end='')

```





## Παράδειγμα 8 – Τα κουμπιά αφής

Η πλακέτα διαθέτει πέντε κουμπιά αφής τα οποία ακουμπώντας τα με το δάχτυλό μας μπορούμε να τα προγραμματίσουμε ώστε να κάνουν όποια ενέργεια θέλουμε. Ακολουθεί ο κώδικας δοκιμής για το 1ο κουμπί. Όταν είναι πατημένο εμφανίζει στο τερματικό το μήνυμα 'Με πάτησες'. Δεν απαιτούνται επιπλέον αρχεία εκτός από το boot.py και το main.py.

```
1 import machine, time
2 threshold = 150 #Κατώφλι. Τιμές κάτω από εδώ σημαίνουν ότι πατήθηκε
3 touch_pin = machine.TouchPad(machine.Pin(12)) #Το πρώτο κουμπί
4
5 print("\nESP32 Touch Demo")
6 while True: # Για πάντα
7     capacitiveValue = touch_pin.read() #Διάβασε τιμές
8     if capacitiveValue < threshold: #Είναι κάτω από το κατώφλι
9         print("Με πάτησες")
10        time.sleep_ms(500)
```

## Παράδειγμα 9 – Σύνδεση WiFi και Web Server

Το άρθρωμα του ESP32 όπως και το ESP8266 διαθέτει υποσύστημα διασύνδεσης WiFi με υποστήριξη πλήρους TCP/IP stack. Στο παράδειγμά μας θα συνδεθεί η πλακέτα με το οικείο σημείο πρόσβασης (Access Point) και με το διαδίκτυο. Θα εκτελέσει έναν μικρό διακομιστή ιστού (WEB) και από τον browser θα αναβοσβήνουμε τα 5 RGB Leds της πλακέτας. Στις γραμμές 4 και 5 βάζουμε το SSID και τον κωδικό του δικού μας WiFi. Δεν απαιτούνται επιπλέον βιβλιοθήκες.

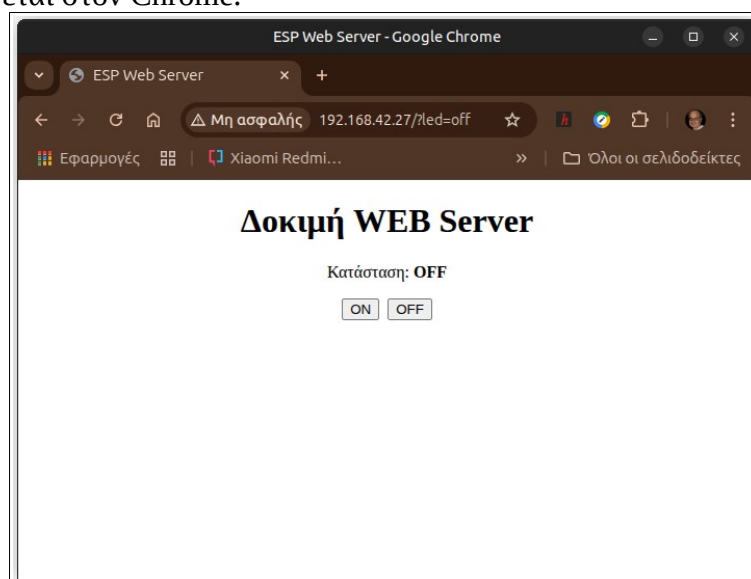
```
1 import machine, time, socket, network, gc, esp, neopixel
2 esp.osdebug(None)
3 gc.collect()
4 ssid = 'SSID' #Το SSID του τοπικού δικτύου WiFi
5 password = 'WiFiPassword' #Ο κωδικός του WiFi
6 station = network.WLAN(network.STA_IF) #θα λειτουργήσει σαν σταθμός ώστε να συνδεθεί στο AP
7
8 station.active(True) #Ενεργοποίηση
9 station.connect(ssid, password) #Σύνδεση
10
11 while station.isconnected() == False: #Περίμενε μέχρι να συνδεθεί
12     pass
13
14 print('Connection successful')
15 print(station.ifconfig())
16
```

```

17 np = neopixel.NeoPixel(machine.Pin(25), 5) #IO25, 5 x RGB Leds
18 led_state = False
19
20 #Ανάβει άσπρα τα 5 Led της πλακέτας
21 def led_ON():
22     global led_state
23     for i in range(5):
24         np[i] = (255, 255, 255) #Άσπρο
25         led_state = True
26         np.write()
27
28 #Σβήνει τα 5 Led της πλακέτας
29 def led_OFF():
30     global led_state
31     for i in range(5):
32         np[i] = (0, 0, 0) #Σβηστό
33         led_state = False
34         np.write()
35
36 #Η ιστοσελίδα που επιστρέφει στον browser
37 def web_page():
38     if led_state == True:
39         cur_state="ON"
40     else:
41         cur_state="OFF"
42     html = """
43 <!DOCTYPE HTML>
44 <html>
45 <head>
46 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
47 <title>ESP Web Server</title>
48 <meta name="viewport" content="width=device-width, initial-scale=1">
49 </head>
50 <body>
51 <center><h1>Δοκιμή WEB Server</h1></center>
52 <p><center>Κατάσταση: <strong>"" + cur_state + ""</strong></center></p>
53 <p><center><a href="/?led=on"><button>ON</button></a>&nbsp;<a href="/?led=off"><button>OFF</button></a></center></p>
54 </body>
55 </html>"""
56     return html
57
58 s = socket.socket() #Δημιουργία υποδοχής TCP (Socket)
59 s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) #Options
60 s.bind(('', 80)) #Ακούει από οποιαδήποτε διεύθυνση IP - Θύρα 80 HTTP
61 s.listen(5) #Ακούει μέχρι 5 ταυτόχρονες συνδέσεις
62
63 while True: #Για πάντα
64     conn, addr = s.accept()
65     print('Got a connection from %s' % str(addr))
66     request = conn.recv(1024) #To αίτημα μέχρι 1024 bytes
67     request = str(request) #Na γίνει string από byte array
68     #print('Content = %s' % request) #Debug
69     request = request.split()[1] #Χωρίς το στα κενά και πάρε το 2ο στοιχείο της λίστας
70     if request == '/?led=on': #Αν υπάρχει αίτημα on
71         print('LED ON')
72         led_ON() #Ανάβει τα Led
73     elif request == '/?led=off': #Διαφορετικά αν υπάρχει αίτημα off
74         print('LED OFF')
75         led_OFF() #Σβήνει τα Led
76     response = web_page() #Ετοιμάσε σελίδα για απάντηση
77     conn.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n') #Επικεφαλίδα HTTP
78     conn.send(response) #Στείλε πίσω στον browser την σελίδα
79     conn.close() #Κλείσε σύνδεση

```

Στις γραμμές 43 έως 56 είναι η σελίδα που εμφανίζεται σε HTML. Στην γραμμή 52 παρεμβάλλεται η μεταβλητή cur\_state ώστε στο σημείο αυτό η ιστοσελίδα να έχει δυναμική συμπεριφορά. Η σελίδα όπως φαίνεται στον Chrome.



## Χρήση ενδιαμέσου κώδικα Byte Code αντί πηγαίου

Η micropython μας δίνει την δυνατότητα εκτέλεσης προμεταγλωττισμένου κώδικα ώστε να επιτύχουμε ταχύτερη εκτέλεση και προστασία του πηγαίου προγράμματος από αντιγραφές. Για να μεταγλωττίσουμε ένα αρχείο χρειάζεται το **mpy\_cross** το οποίο το εγκαθιστούμε στον υπολογιστή μας με **pip install mpy\_cross**.

Αν θέλουμε να μεταγλωττίσουμε το αρχείο **main.py** πάμε στον φάκελο όπου βρίσκεται και γράφουμε **python -m mpy\_cross main.py** και μας δημιουργεί το αρχείο **main.mpy**.

Το ανεβάζουμε από το **vscode - pymakr** στην πλακέτα και σβήνουμε το πηγαίο **main.py**. Τέλος στο **boot.py** γράφουμε σε μια γραμμή **import main**.

## Παράδειγμα 10 – Ο αισθητήρας χειρονομίας

Η πλακέτα διαθέτει τον αισθητήρα PAJ7620U2 ο οποίος ανιχνεύει έως 9 χειρονομίες. Κουνώντας το χέρι μας σε απόσταση μέχρι 10 εκατοστά πάνω από τον αισθητήρα, αυτός καταλαβαίνει την χειρονομία που κάναμε. Με αυτό τον τρόπο μπορούμε να δώσουμε εντολές σε ένα σύστημα μόνο με την κίνηση του χεριού μας. Η σωστή διεύθυνση του αισθητήρα είναι η 0x73 και όχι η 0x68 που αναφέρεται πάνω στην πλακέτα. Μέσα στο σύστημα αρχείων πρέπει να συμπεριλάβουμε και το **module gesture.py**.

```

1 import machine, time
2 i2c = machine.I2C(0, sda = machine.Pin(21), scl = machine.Pin(22), freq = 400000)
3 #Η εναλλακτικά
4 #i2c = machine.SoftI2C(sda=machine.Pin(21), scl=machine.Pin(22))
5
6 from gesture import Gesture
7 g=Gesture(i2c)
8 #
9 # 0: Τίποτα
10 # 1: Μπροστά
11 # 2: Πίσω
12 # 3: Δεξιά
13 # 4: Αριστερά
14 # 5: Πάνω
15 # 6: Κάτω
16 # 7: Περιστροφή σύμφωνα με τους δείκτες του ρολογιού
17 # 8: Περιστροφή αντίθετα με τους δείκτες του ρολογιού
18 # 9: Κυματισμός
19
20 while 1:
21     #g.print_gesture() #Εμφανίζει την χειρονομία
22     value=g.return_gesture()
23     #print(value) #Debug
24     #if value==0:
25     #    print("Αδράνεια") # nothing
26     if value==1:
27         print("Μπροστά") #Το χέρι κινείται από πάνω προς τον αισθητήρα (άξονας Z)
28     if value==2:
29         print("Πίσω") #Το χέρι κινείται από τον αισθητήρα προς τα πάνω (άξονας Z)
30     if value==3:
31         print("Δεξιά") #Δεξιά (άξονας X)
32     if value==4:
33         print("Αριστερά") #Αριστερά (άξονας X)
34     if value==5:
35         print("Πάνω") #Πάνω (άξονας Y)
36     if value==6:
37         print("Κάτω") #Κάτω (άξονας Y)
38     if value==7:
39         print("Δείκτες ρολογιού") #Περιστροφή (άξονας X-Y)
40     if value==8:
41         print("Αντίθετα από δείκτες ρολογιού") #Αντίθετη περιστροφή (άξονας X-Y)
42     if value==9:
43         print("Κυματισμός") #Γρήγορα πάνω κάτω άξονα Y
44     time.sleep(1)

```

## Παράδειγμα 11 – Η κάρτα μνήμης SD

Μπορούμε να αποθηκεύουμε ή να διαβάζουμε μεγάλα αρχεία όπως εικόνες, μουσική, αρχεία καταγραφής κλπ. στην εξωτερική κάρτα μνήμης SD. Θα χρειαστούμε μια καρτούλα microSD χωρητικότητας έως 32GB (ίσως υποστηρίζει και μεγαλύτερες). Το σύστημα αρχείων μπορεί να είναι FAT32 ή LittleFS. Για συμβατότητα ώστε να διαβάζεται από PC θα χρησιμοποιήσουμε σύστημα FAT32. Σε μνήμη 32GB η μονάδα εκχώρησης (allocation unit) είναι 8192 bytes. Η κάρτα συνδέεται με την διεπαφή SPI. Δεν θα χρειαστούμε επιπλέον αρχεία modules στο σύστημα αρχείων.

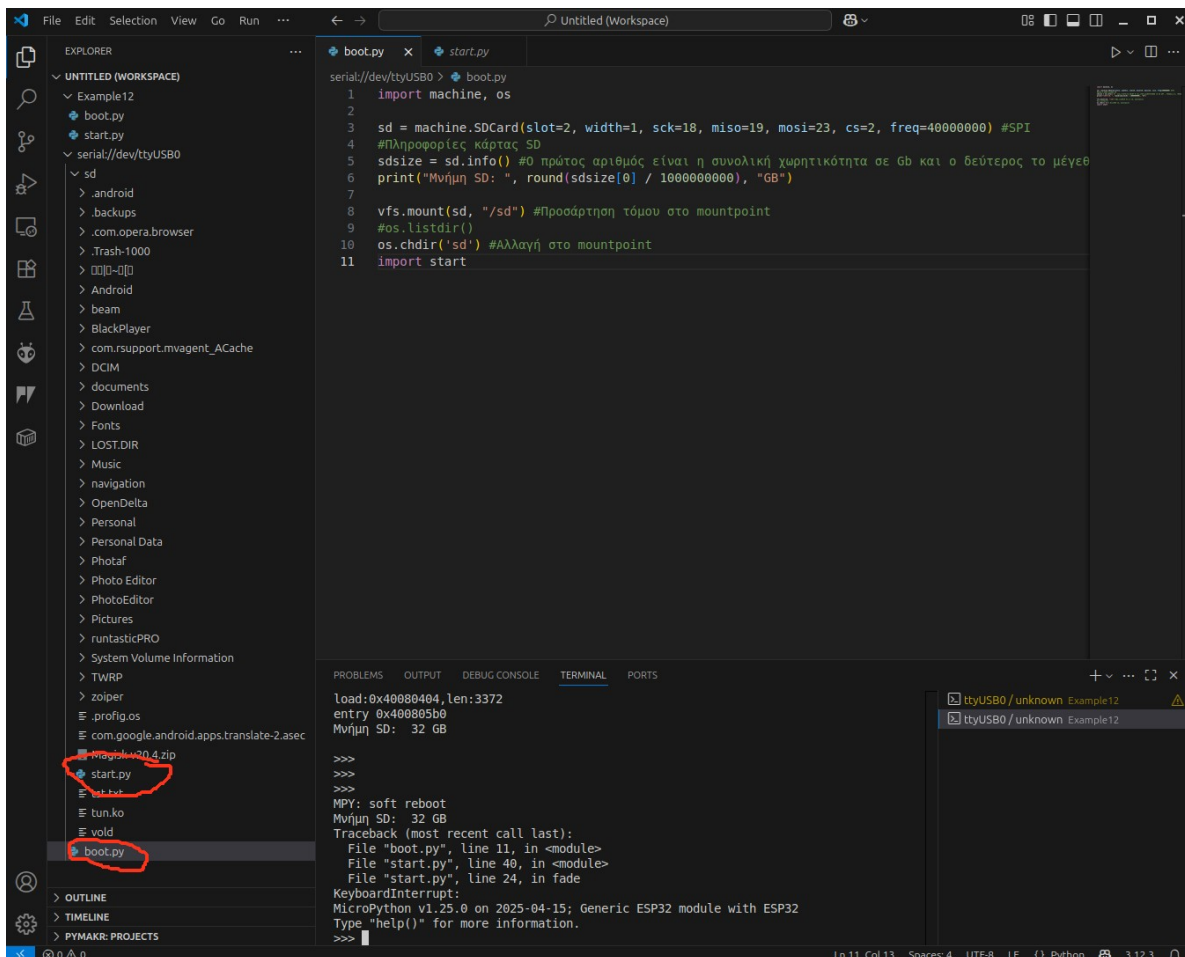
```

1 import machine, os
2
3 sd = machine.SDCard(slot=2, width=1, sck=18, miso=19, mosi=23, cs=2, freq=40000000) #SPI
4 #Πληροφορίες κάρτας SD
5 sdsdsize = sd.info() #0 πρώτος αριθμός είναι η συνολική χωρητικότητα σε Gb και ο δεύτερος το μέγεθος τομέα (512)
6 print("Μνήμη SD: ", round(sdsdsize[0] / 1000000000), "GB")
7
8 vfs.mount(sd, "/sd") #Προσάρτηση τόμου στο mountpoint
9 stat = os.statvfs('/sd') #Πληροφορίες κατάτμησης
10 print("Συνολικός χώρος κατάτμησης:", round(stat[0] * stat[2] / 1000000000, 1), "GB")
11 print("Σε χρήση:", round(stat[0] * (stat[2] - stat[3]) / 1000000000, 1), "GB")
12
13 os.chdir('/sd') #Αλλαγή στο mountpoint
14 print(os.listdir()) #Εμφάνιση αρχείων καταλόγου
15 #vfs.umount('/sd') #Αποπροσάρτηση τόμου

```

## Παράδειγμα 12 – Εκτέλεση προγράμματος από την κάρτα SD

Υπάρχει η δυνατότητα εκτέλεσης προγραμμάτων Python τα οποία είναι αποθηκευμένα στην κάρτα μνήμης SD. Στον κατάλογο root της εσωτερικής μνήμης Flash αποθηκεύουμε μόνο το αρχείο **boot.py** το οποίο στην γραμμή 3 αρχικοποιεί την επικοινωνία της SD μέσω του διαύλου SPI και στην γραμμή 8 κάνει προσάρτηση του διαμερίσματος της SD στον κατάλογο /sd. Στην 10 μπαίνουμε στον κατάλογο sd και τέλος στην 11 κάνουμε import το αρχείο start.py ώστε να εκτελεστεί. Το αρχείο **start.py** έχει αποθηκευτεί στην μνήμη SD και είναι αυτό του παραδείγματος 3 (Τροποποίηση).



Στην παραπάνω εικόνα φαίνεται ένα διαμέρισμα FAT32 από κάρτα SD που είχε Android και δεν έχει γίνει format. Το αρχείο start.py αντιγράφεται στο root του /sd. Ακολουθεί το αρχείο boot.py

```

1 import machine, os
2
3 sd = machine.SDCard(slot=2, width=1, sck=18, miso=19, mosi=23, cs=2, freq=40000000) #SPI
4 #Πληροφορίες κάρτας SD
5 sdsdsize = sd.info() #Ο πρώτος αριθμός είναι η συνολική χωρητικότητα σε Gb και ο δεύτερος το μέγεθος τομέα (512)
6 print("Μνήμη SD: ", round(sdsdsize[0] / 1000000000), "GB")
7
8 vfs.mount(sd, "/sd") #Προσάρτηση τόμου στο mountpoint
9
10 os.chdir('sd') #Αλλαγή στο mountpoint
11 import start

```

### Παράδειγμα 13 – Ο αισθητήρας κίνησης 6 αξόνων (MEMS) MPU6050

Η πλακέτα διαθέτει και το chip **MPU6050** το οποίο συνδέεται μέσω του διαύλου I2C στην διεύθυνση 0x68. Ο αισθητήρας MPU6050 είναι ένα chip με μια μονάδα **MEMS** (Micro Electro-Mechanical System). Αυτό το τσιπ περιέχει γυροσκόπιο τριών αξόνων, επιταχυνσιόμετρο τριών αξόνων και ψηφιακό επεξεργαστή ελέγχου κίνησης. Επιπλέον, περιέχει έναν ενσωματωμένο αισθητήρα θερμοκρασίας. Μπορούμε να χρησιμοποιήσουμε αυτήν τη μονάδα για τη μέτρηση της ταχύτητας, της επιτάχυνσης, του προσανατολισμού, της μετατόπισης και άλλων παραμέτρων που σχετίζονται με την κίνηση. Σήμερα, όλα τα σύγχρονα smartphones διαθέτουν ενσωματωμένο αισθητήρα αδρανειακής κίνησης. Αυτός ο αισθητήρας παρέχει μια ολοκληρωμένη λύση για οποιοδήποτε σύστημα παρακολούθησης κίνησης έξι αξόνων.

Για το παράδειγμα θα χρειαστούμε και το αρχείο MPU6050.py να βρίσκεται μέσα στο root του συστήματος αρχείων.

```

1 from MPU6050 import MPU6050
2
3 from machine import Pin
4 from time import sleep_ms
5
6 mpu = MPU6050()
7
8 # Για πάντα
9 while True:
10     # Επιταχυνσιόμετρο
11     '''
12     accel = mpu.read_accel_data() # Διάβασε το επιταχυνσιόμετρο [m / s^2]
13     aX = accel["x"]
14     aY = accel["y"]
15     aZ = accel["z"]
16     print("x: " + str(aX) + " y: " + str(aY) + " z: " + str(aZ))
17     '''
18
19     # Γυροσκόπιο
20     '''
21     gyro = mpu.read_gyro_data() # Διάβασε το γυροσκόπιο [deg/s]
22     gX = gyro["x"]
23     gY = gyro["y"]
24     gZ = gyro["z"]
25     print("x: " + str(gX) + " y: " + str(gY) + " z: " + str(gZ))
26     '''
27
28     # Θερμοκρασία
29     temp = mpu.read_temperature() # Διάβασε την θερμοκρασία της συσκευής [degC]
30     print("Θερμοκρασία: " + str(temp) + "°C")
31
32     # G-Force
33     gforce = mpu.read_accel_abs(g=True) # Διάβασε το μέτρο της επιτάχυνσης
34     print("G-Force: " + str(gforce))
35
36     # Χρόνος παύσης σε (ms)
37     sleep_ms(100)

```

Στο παραπάνω παράδειγμα δοκιμάζουμε το επιταχυνσιόμετρο, γυροσκόπιο, θερμόμετρο και την δύναμη G του αισθητήρα. Τα δεδομένα εμφανίζονται στο τερματικό. Σε επόμενο παράδειγμα θα εμφανίζουμε την κλίση της πλακέτας τριςδιάστατα με γραφικό τρόπο.