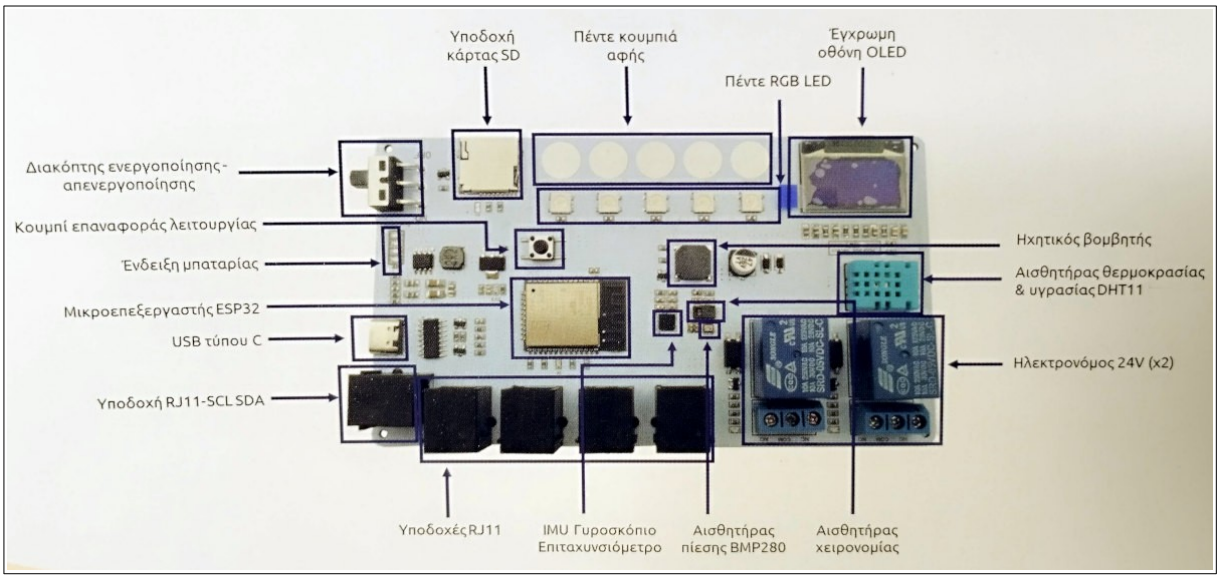
MicroPython στο ARD:Icon II

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

……

Χρήση του IDE VS Code με την Micropython

……

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           cmath             json              struct
_onewire          collections       machine           sys
_thread           cryptolib         math              time
_webrepl          deflate           micropython       tls
aioespnow         dht               mip/__init__      uasyncio
apa106            ds18x20           neopixel          uctypes
array             errno             network           umqtt/robust
asyncio/__init__  esp               ntptime           umqtt/simple
asyncio/core      esp32             onewire           upysh
asyncio/event     espnow            os                urequests
asyncio/funcs     flashbdev         platform          vfs
asyncio/lock      framebuf          random            webrepl
asyncio/stream    gc                re                webrepl_setup
binascii          hashlib           requests/__init__ websocket
Plus any modules on the filesystem
```

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_WAKE', 'Timer', '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, 0, 255)
```

Δηλαδή ο συνολικός χώρος είναι 4096 x 512 = 2097152 = 2 MB και ο ελεύθερος χώρος είναι 4096 x 483 = 1978368 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
```

Το πρώτο μας πρόγραμμα - blink

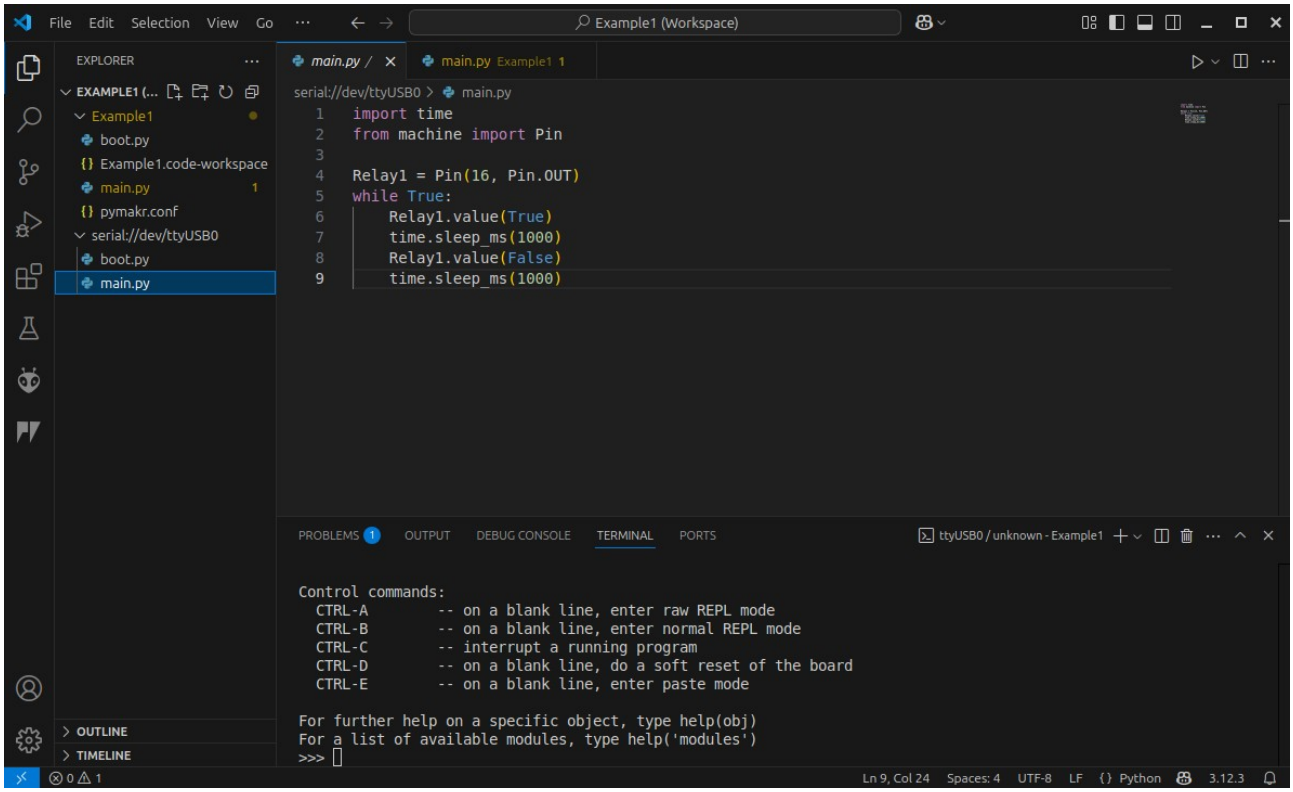
Θα γράψουμε ένα πολύ απλό πρόγραμμα που δεν είναι άλλο από το blink του Arduino αλλά σε γλώσσα Python. Φτιάχνουμε ένα νέο Project με όνομα Example1 και το αποθηκεύουμε σε κάποιο φάκελο. Επιλέγουμε συσκευή com της πλακέτας και γράφουμε στο αρχείο main.py τον παρακάτω κώδικα:

```
import time
from machine import Pin

Relay1 = Pin(16, Pin.OUT)
while True:
    Relay1.value(True)
    time.sleep_ms(1000)
```

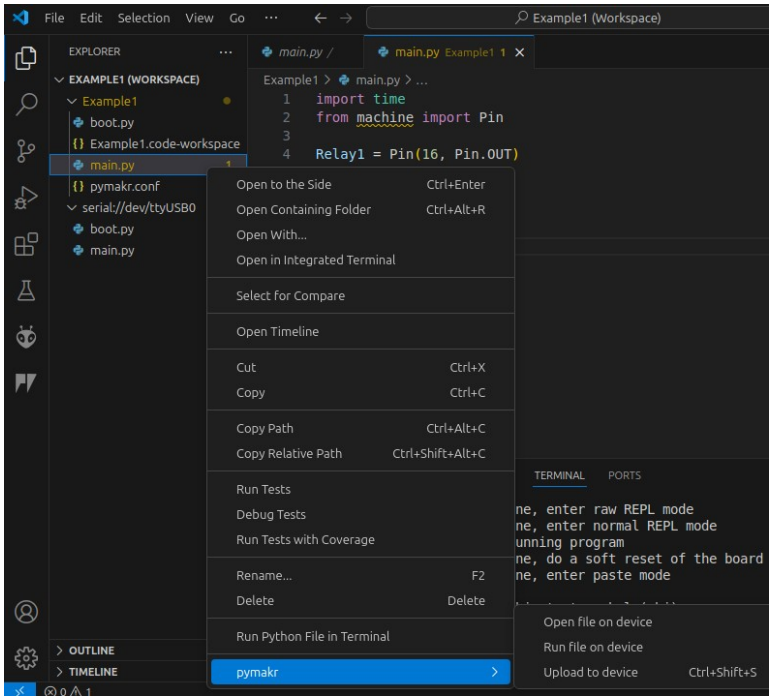
```
    Relay1.value(False)
    time.sleep_ms(1000)
```

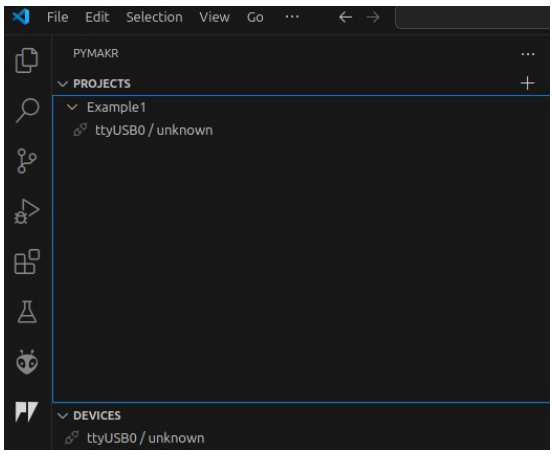
Το πρώτο Relay συνδέεται στο IO16. Στην 3η γραμμή γίνεται έξοδος και μέσα στην while η οποία εκτελείται για πάντα αρχικά το ενεργοποιεί, περιμένει 1sec, το απενεργοποιεί και περιμένει άλλο ένα sec.
Όπως βλέπουμε στον Explorer αριστερά έχουμε τον τοπικό φάκελο του έργου και το σύστημα αρχείων της συσκευής. 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.



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.





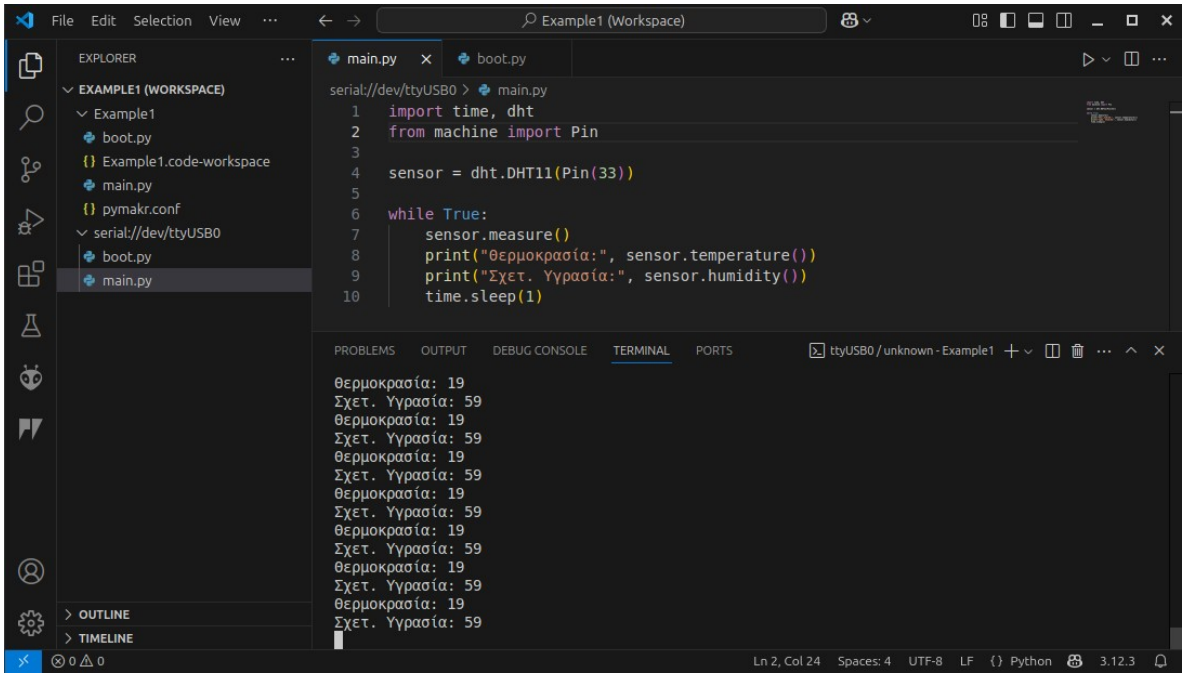
Παράδειγμα 2 – Μέτρηση υγρασίας και θερμοκρασίας με το DHT11

Γράφουμε τον παρακάτω κώδικα στο main.py.

```
import time, dht
from machine import Pin

sensor = dht.DHT11(Pin(33))

while True:
    sensor.measure()
    print("Θερμοκρασία:", sensor.temperature())
    print("Σχετ. Υγρασία:", sensor.humidity())
    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.

```
import machine, neopixel

np = neopixel.NeoPixel(machine.Pin(25), 5) #IO25, 5 x RGB Leds
#         R , G , B
np[0] = (255, 0, 0) #Κόκκινο 100%
np[1] = (125, 204, 223)
np[2] = (120, 153, 23)
np[3] = (255, 0, 153)
np[4] = (0, 0, 64) #Μπλε 25%
np.write()
```

Τροποποίηση

Να τροποποιηθεί ο παραπάνω κώδικας ώστε να κάνει διάφορα οπτικά effects όπως fade in, fade out, Kit κλπ.

```
import machine, neopixel
import time

np = neopixel.NeoPixel(machine.Pin(25), 5) #IO25, 5 x RGB Leds

def blank():
    np[0] = np[1] = np[2] = np[3] = np[4] = (0, 0, 0)
    np.write()

def fade(R = 0, G = 0, B = 0):
    #Fade in
    for i in range(0, 256, 5):
        r = i * R; g = i * G; b = i * B
        for j in range(5):
            np[j] = (r, g, b)
        np.write()
        time.sleep_ms(50)
    #Fade out
    for i in range(255, -1, -5):
        r = i * R; g = i * G; b = i * B
        for j in range(5):
            np[j] = (r, g, b)
        np.write()
        time.sleep_ms(50)

def kit(color, times):
    for k in range(times):
        for i in range(5):
            np[0] = np[1] = np[2] = np[3] = np[4] = (0, 0, 0)
            np[i] = color
            np.write()
            time.sleep_ms(100)
        for i in range(4, -1, -1):
            np[0] = np[1] = np[2] = np[3] = np[4] = (0, 0, 0)
            np[i] = color
            np.write()
            time.sleep_ms(100)

while(True):
    fade(1, 0, 0)
    fade(0, 1, 0)
    fade(0, 0, 1)
    kit((0, 255, 128), 5)
    kit((255, 0, 128), 5)
    kit((255, 0, 0), 5)
    blank()
    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.

```
import time
from machine import Pin

beeper = Pin(5, Pin.OUT) #Ο βομβητής συνδέεται στο pin IO5

def beep():
    for i in range(200):
        beeper.value(1)
        time.sleep_us(1000)
        beeper.value(0)
        time.sleep_us(1000)

beep()
```

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

```
import time
from machine import Pin

beeper = Pin(5, Pin.OUT) #Ο βομβητής συνδέεται στο pin IO5

def beep(freq = 500, dur = .5): #Default values
    #Υπολογισμοί
    period = 1.0 / freq #Περίοδος
    half_per = int((period / 2) * 1000000) #Ημιπερίοδος
    times = int(dur / period) #Αριθμός κύκλων
    for i in range(times):
```

```
        beeper.value(1)
        time.sleep_us(half_per)
        beeper.value(0)
        time.sleep_us(half_per)

beep() #Θα παραχθεί τόνος 500Hz για 0,5sec
beep(1000, .5) #1000Hz
beep(2000, .5) #2000Hz

for i in range(200, 4001, 10): #Σταδιακό ανέβασμα από 200 – 4000
    beep(i, .003)
for i in range(4000, 195, -10): #Σταδιακό κατέβασμα από 4000 – 200
    beep(i, .003)
```

Παράδειγμα 5 – Ο αισθητήρας BMP280

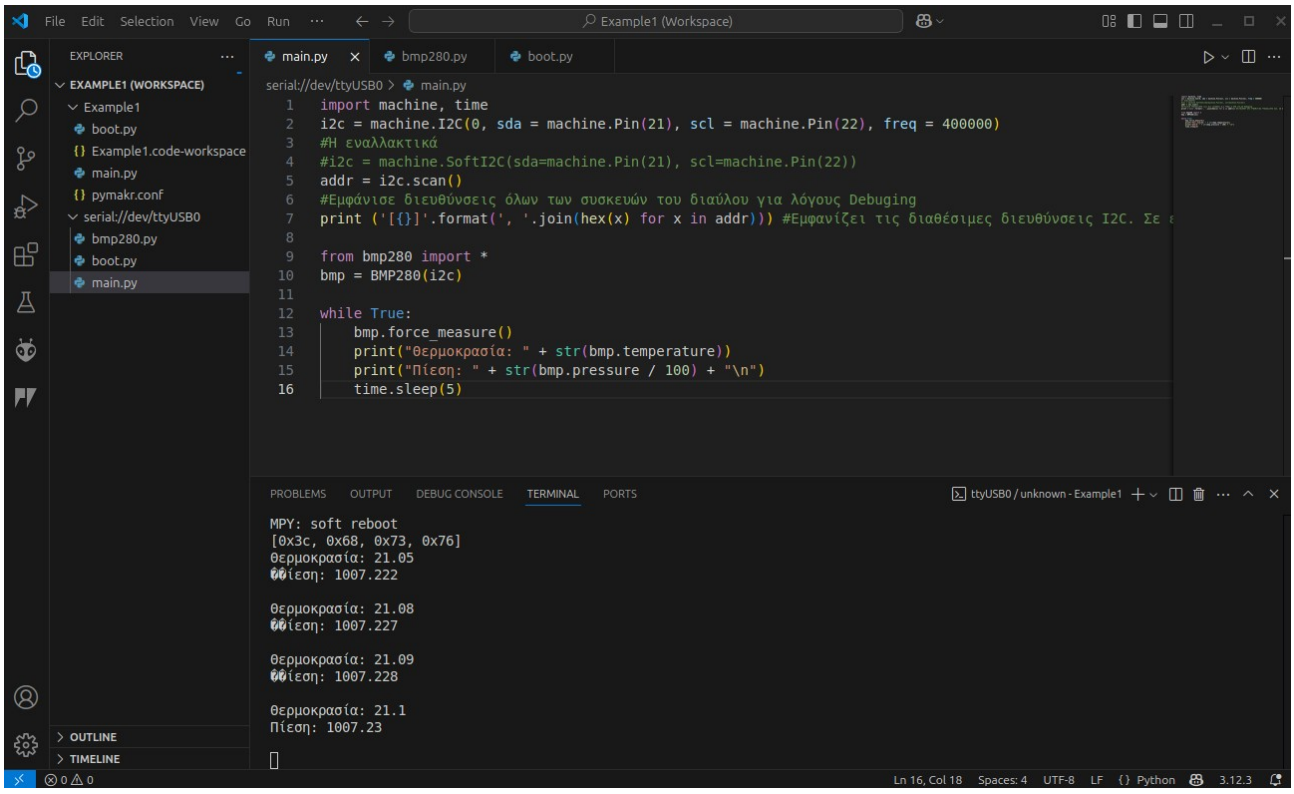
Ο αισθητήρας BMP280 της BOSCH είναι ενσωματωμένος στην πλακέτα και χρησιμοποιείται για μέτρηση ατμοσφαιρικής πίεσης και θερμοκρασίας. Συνδέεται με τον ESP32 μέσω του σειριακού διαύλου IIC ή I^2C. Επειδή δεν είναι ενσωματωμένο το module στην Micropython θα πρέπει να το συμπεριλάβουμε στο σύστημα αρχείων 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 στην διεύθυνση https://srv1-1sek-prevez.pre.sch.gr/openeclass/modules/video/file.php?course=PLIROFORIKI101&id=36 .

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.

```
import machine, time
i2c = machine.I2C(0, sda = machine.Pin(21), scl = machine.Pin(22), freq = 400000)
#Η εναλλακτικά
#i2c = machine.SoftI2C(sda=machine.Pin(21), scl=machine.Pin(22))
addr = i2c.scan()
#Εμφάνισε διευθύνσεις όλων των συσκευών του διαύλου για λόγους Debuging
print ('[{}]'.format(', '.join(hex(x) for x in addr))) #Εμφανίζει τις διαθέσιμες διευθύνσεις I2C. Σε εμάς είναι η 0x76

from bmp280 import *
bmp = BMP280(i2c)

while True:
    bmp.force_measure()
    print("Θερμοκρασία: " + str(bmp.temperature))
    print("Πίεση: " + str(bmp.pressure / 100) + "\n")
    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 OLED

Η πλακέτα διαθέτει οθόνη OLED 0,96’’ και ανάλυσης 128x64 pixels. Η οθόνη συνδέεται στον ίδιο δίαυλο 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 στην διεύθυνση https://srv1-1sek-prevez.pre.sch.gr/openeclass/modules/video/file.php?course=PLIROFORIKI101&id=37 .

```
import machine, time
i2c = machine.I2C(0, sda = machine.Pin(21), scl = machine.Pin(22), freq = 400000)
#Η εναλλακτικά
#i2c = machine.SoftI2C(sda=machine.Pin(21), scl=machine.Pin(22))
addr = i2c.scan()
#Εμφάνισε διευθύνσεις όλων των συσκευών του διαύλου για λόγους Debuging
print ('[{}]'.format(', '.join(hex(x) for x in addr))) #Εμφανίζει τις διαθέσιμες διευθύνσεις I2C. Σε εμάς είναι η 0x76

from ssd1306 import SSD1306_I2C
#display = SSD1306_I2C(128, 64, i2c) #Λίπλα μπορώ να βάλω διεύθυνση π.χ. SSD1306_I2C(128, 64, i2c, 0x3c)
display = SSD1306_I2C( width=128, height=64, i2c=i2c, addr=0x3c, external_vcc=False )
display.text('Hello', 5, 5)  #Θέση x, y
display.text('World', 5, 15, 1) #color 1/0
display.rect(0,0,127,63,1,0) #x, y, width, heigh, color=0/1, fill=0/1
display.show()

#display.fill(0) #Καθαρίζει οθόνη
#display.show()
```

LCD character

```
# ------- I2C  Address -----
I2C_Addr = 0x27 #  address may be changed by soldering connections on PCF8574 backpack

# ------- LCD Dimension ----
LCD_Dim = (16, 2)

from machine import I2C,  Pin
i2c = I2C(0, sda=Pin(21), scl=Pin(22), freq=100000)


import gc
from time import sleep
from lcd_i2c8574 import I2cLcd

lcd = I2cLcd(i2c, I2C_Addr, LCD_Dim)
gc.collect(); mfree1 = gc.mem_free()

lcd.clear()
lcd.write('Hello World', end='')
```

Touch Sensor

```
capacitiveValue = 500
threshold = 150 # Threshold to be adjusted
touch_pin = machine.TouchPad(machine.Pin(12))

print("\nESP32 Touch Demo")
while True: # Infinite loop
  capacitiveValue = touch_pin.read()
  if capacitiveValue < threshold:
    print("Με πάτησες")
    time.sleep_ms(500)
```