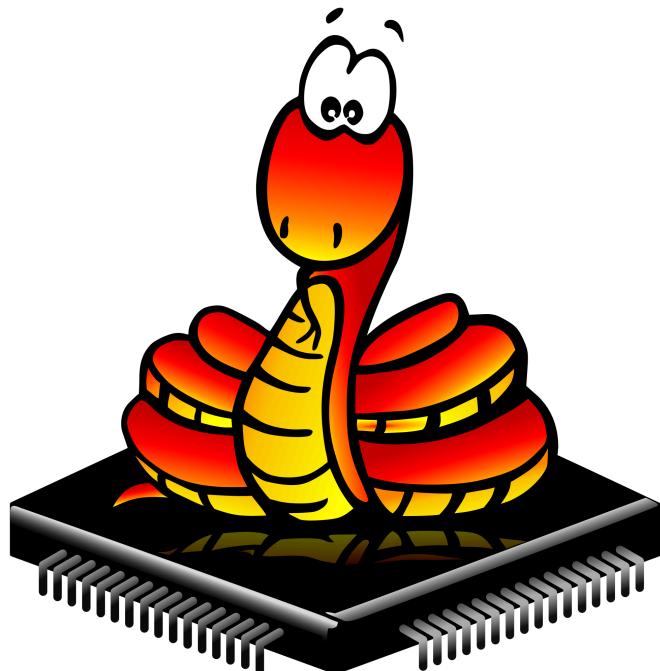


תכנות MicroPython למורי הנדסת אלקטרוני ומחשבים



מדריך למורה

מאי 2025

גרסה 4.26

כתב: גדי הרמן

תוכן עניינים

4	משימה 1 - התקנת MicroPython על גבי בקר ESP32 וסביבת הפיתוח Thonny
16	משימה 2 - כתיבת קוד MicroPython בסביבת הפיתוח Visual Studio Code
21	משימה 3 - כתיבת תוכנית ראשונה לביוץ פלט בשפת Python
29	משימה 4 - שימוש ב- Timer פנימי
35	משימה 5 - פסיקות חומרה
40	משימה 6 - קלט אות אנלוגי
47	משימה 7 - ממיר דיגיטלי לאנלוגי DAC
56	משימה 8 - הפעלת מד מרחוק אולטראסוני דגם hc-sr04
62	משימה 9 - תקשורת UART בין 2 בקרים
69	משימה 10 - תקשורת Bluetooth מבוססת HC-05 HC-06 או HC-08
81	משימה 11 - תקשורת Bluetooth Low Energy בברker ESP32
96	משימה 12 - אתחול קישוריות ה- WiFi בברker ESP32
1	משימה 13 - מימוש שרת אינטרנט מבוסס HTTP
1	משימה 14 - מימוש שרת אינטרנט מבוסס HTTP GET
1	משימה 15 - הפעלת צג גרפי דגם SSD1306 OLED display
1	משימה 16 - שירות ענן מבוססי MQTT
1	משימה 17 - הפעלת רכיב השמעת קבצי MP3 מבוסס על YX5300
1	משימה 18 - הפעלת צג גרפי 2x16 I2C LCD
1	משימה 19 - הפעלת צג LCD גרפי צבעוני 320*240 פיקסלים מבוסס על ILI9341
1	משימה 20 - קריאה וכתיבה של תגיות RFID תוך שימוש ב- RC522
1	משימה 21 - עדכון RTC פנימי בברker תוך שימוש ב- API מבוסס JSON
1	משימה 22 - תקשורת אלחוטית מבוססת מקם"ש NRF24L01
1	משימה 23 - חיישני משקל המבוסס על ממיר HX711
1	משימה 24 - שירות העברת מסרים מבוסס Telegram
1	משימה 25 - חיישן טביעה אצבע - DY50
1	משימה 26 - מערכת הקבצים של בקר ESP32
1	משימה 27 - תבנית לפרויקט המשלב מערכת ניהול משתמשים (תוך שימוש בהצפנה!!!)
1	משימה 28 - תאורות רצף נורות לד - NeoPixel
1	משימה 29 - הציגת טקסט על גבי מטריצות 8x8 NeoPixel ב- ESP32 עם MicroPython
1	משימה 30 - עבודה עם ערוץ תקשורת I2C - ESP32 ועבודה עם חיישנים
1	נספח א' - בדיקת הספריות הזמיןות לתוכנות ב- MicroPython תחת בקר ESP32
1	נספח ב' - יסודות בתוכנות אסינכרוני ב- Thonny
1	נספח ג' - יסודות בתוכנות אסינכרוני ב- MicroPython Thonny
1	נספח ד' - יבוא ספריות קוד ייעודיות ל- MicroPython
1	נספח ה' - מיפוי הדקי בברker ESP32
1	נספח ו' - עדכון קושחה לבקר ESP32
1	תנאי השימוש

מורים יקרים,

במטרה לקדם את החינוך הטכנולוגי בכלל ובמגמת הנדסת אלקטרוניתקה ומחשבים בפרט כתיבתי מדריך למורה שיעזר לכם להיכנס לעולם מיקרו-בקרים הניתנים לתוכנות בשפת חספota MicroPython.

לרשוכם פתחתי קבוצה Whatsapp בשם "קהילת מורי MicroPython בבר" ESP32" שתשתמש כפלטפורמה לעזרה ושאלות בתחום זה. אתם מוזמנים ללמוד מהמדריך הנ"ל ובמקביל לפרסום ולשאול שאלות בקבוצה.

להלן הקישור לקבוצה:

<https://chat.whatsapp.com/LfOH8x5RSI6GKWSSePUAAb>

ניתן להוריד את כל קבצי הקוד בספר כמו כן את הגרסה الأخيرة של ספר זה דרך הקישור הבא:

https://github.com/GadiHerman/ESP32_MicroPython_AllBookFiles

בברכה

גדי הרמן

משימה 1 - התקנת Thonny על גבי בקר ESP32 וסביבת הפיתוח MicroPython

קישורים:

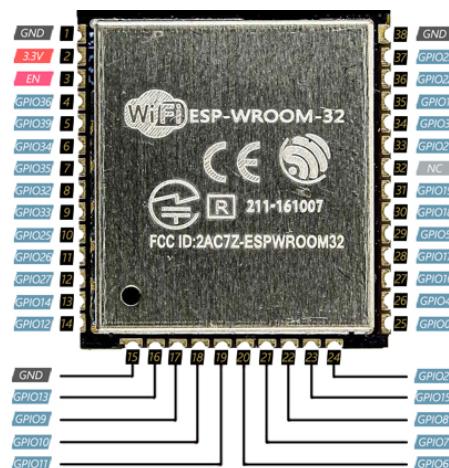
<https://www.youtube.com/watch?v=GJinEfhD4Cw>

היכרות עם בקר ESP32

ESP32 הוא בקר רב עצמה מוגן SoC(Clomer System on Chip) ממערכת מחשב מלאה על גבי רכיב אחד. שפותה על ידי חברת סינית בשם Espressif Systems.

<https://www.espressif.com/en/products/socs/esp32>

סוגים שונים של מיקרו-בקרים ESP32 זמינים בשוק. במדריך זה אני מתמקד בגרסת-32 ESP-WROOM-32, הידוע גם בשם WROOM32. מיקרו-בקר זה זוכה לשבחים רבים ונפוץ בהרבה לוחות פיתוח מבוססי ESP32 כגון ESP32 DEVKIT. גם אם אתם משתמשים בלוחות פיתוח אחרים המבוססים על מיקרו-בקר ESP32, מדריך זה עשוי להיות רלוונטי גם לכם, שכן מיקרו-בקרים אחרים של ESP32 חולקים תכונות הדקים דומות ל-WROOM32. לבקר זה 48 הדקים, כל אחד מהם משמש לפונקציות רבות. אך לא כל ההדקים נגישים בכל לוח פיתוח של ESP32, ולהדקים מסוימים עשויים להיות מגבלים שימושם. להלן, תיאור בסיסי של הדקי GPIO הזמינים בברker ESP32-WROOM-32:



הברקרים הנ"ל פופולריים מאוד ומשמשים בעיקר ליישומי IoT כЛОם ליישומי "ה האינטרנט של הדברים" - .internet of things

הברker מספק ביצועים גבוהים בפרויקטים הדורשים עבודה עם מעבד (MCU) בעל 2 ליבות הכלול תקשורת רבתות ועבודה בזרמים נמוכים לאורך זמן. בנוסף, ESP32 מספק ביצועים גבוהים עם MCU ליבה כפולה העודדת בתדר של 200MHz כמו הברker מצויד בזיכרון הבזק בנפח של 4 מגה-בייט. הברker מצויד ברוב הממשקים הזמינים היום כולל Wi-Fi BT I2C SPI LCD I2S GPS.

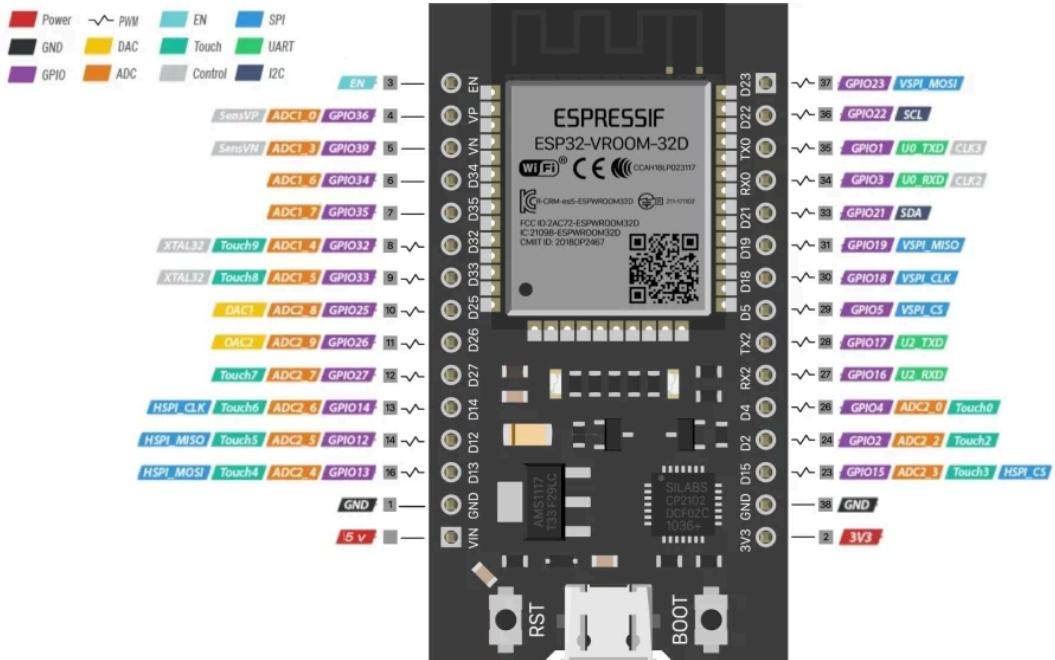
הברker מגיע עם מערכת ההדקים הבא:

- 18 12-bit ADC pins
- 2 8-bit DAC pins
- 3 SPI interfaces
- 2 I2C interfaces
- 3 UART interfaces
- 16 PWM channels

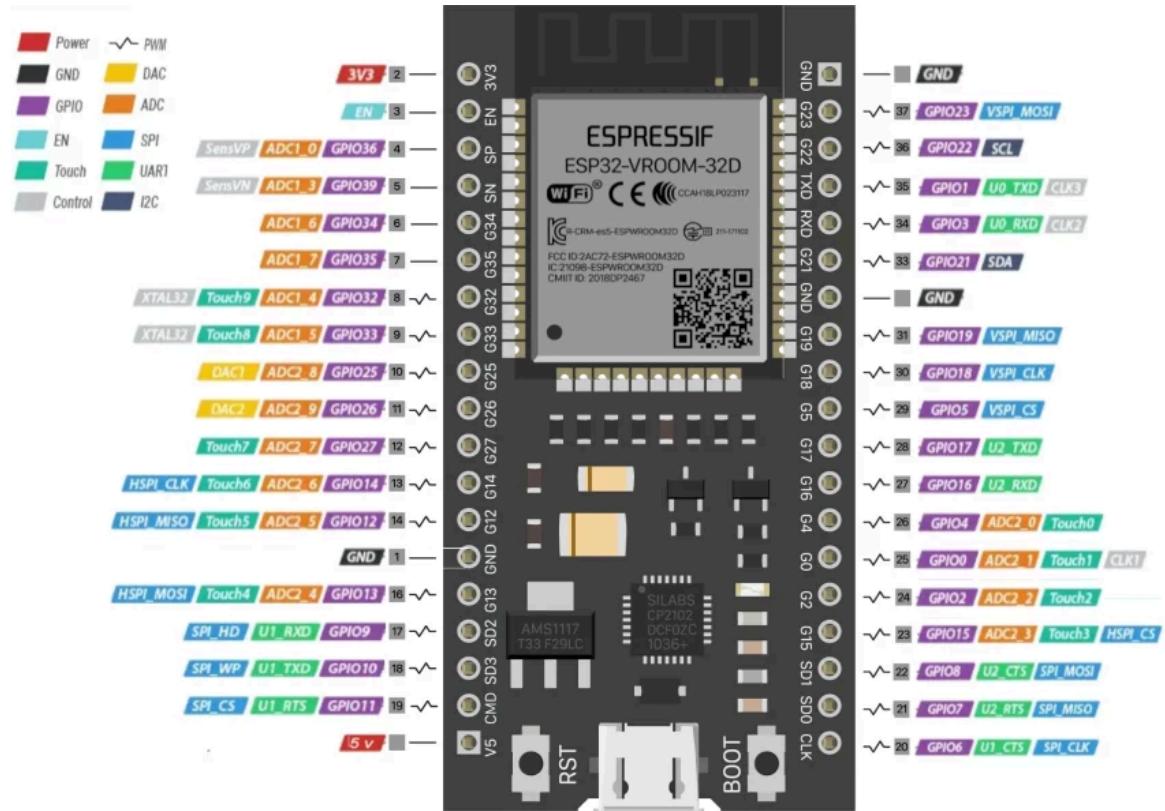
- 10 touch pins

לצורך עבודת פיתוח תוכנה תחת בקר ESP-WROOM-32 ESP-Chip חבותות שונות מספקות מספר מוצרים. ערכת הפיתוח הופולארית ביותר היא ESP32-DevkitC הנמכרת עלות של כ- 4 עד 6 דולר ליחידה.

מספר לוחות פיתוח המבוססים על ESP32-DevkitC זמינים עם הקצאות פינים שונות. לא כל פין ה-GPIO קיימם בכל הלוחות, אולם כל ה-GPIO מתנהגים אותו הדבר, לא משנה באיזה לוח פיתוח ESP32 נעשה שימוש. ניתן לראות את הקצאת הפינים של גרסת 30 הפינים של לוח הפיתוח ESP32-WROOM-32D בתמונה למטה.



האיור הבא מציג את הקצאת הפינים של גרסת 38 הפינים של לוח הפיתוח ESP32.



כרטיס הפיתוח מספק לנו את כל מה שהוא צריך לו כדי לתוכנת את הבקר, דרך ממשק USB המאפשר לחבר את הבקר לחשב PC ולעבוד ישירות אליו.

ניתן לתוכנת את הבקר במספר שפות פיתוח. במסגר זה נמקד את הדרישה שלנו בפיתוח תוכנה בסביבת Python או ליתר דיוק בסביבת MicroPython שהיא גרסה רזה של שפת Python המותאמת לעובדה על מספר בקרים בניהם ESP32.

התקנת סביבת הפיתוח Thonny

Thonny היא סביבת פיתוח המיעדרת למתחילהם. הסביבה מאפשרת ריצת קוד בשפת Python, בשפת MicroPython. כמו כן סביבת הפיתוח זו תעזר לנו להתקין את הקושחה הרלוונטיות כדי להפוך את ה-ESP32 לעבודה עם MicroPython. להלן קישור לאתר התוכנה:

<https://thonny.org/>



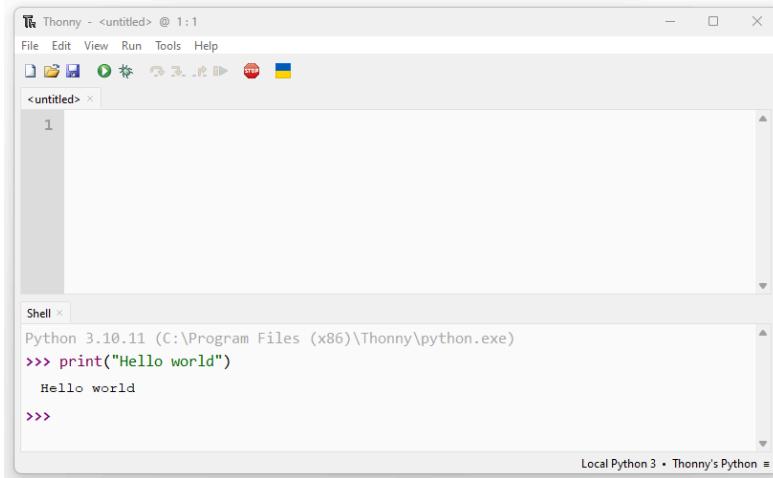
Official downloads for Windows

Installer with 64-bit Python 3.10, requires 64-bit Windows 8.1 / 10 / 11
[thonny-4.1.4.exe \(21 MB\)](#) ↗ recommended for you

Installer with 32-bit Python 3.8, suitable for all Windows versions since 7
[thonny-py38-4.1.4.exe \(20 MB\)](#)


Download version **4.1.4** for
Windows • Mac • Linux

נריץ את סביבת הפיתוח:



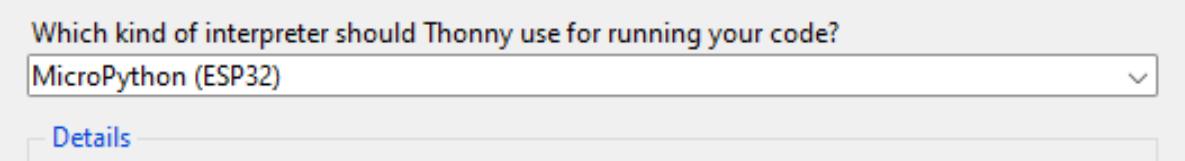
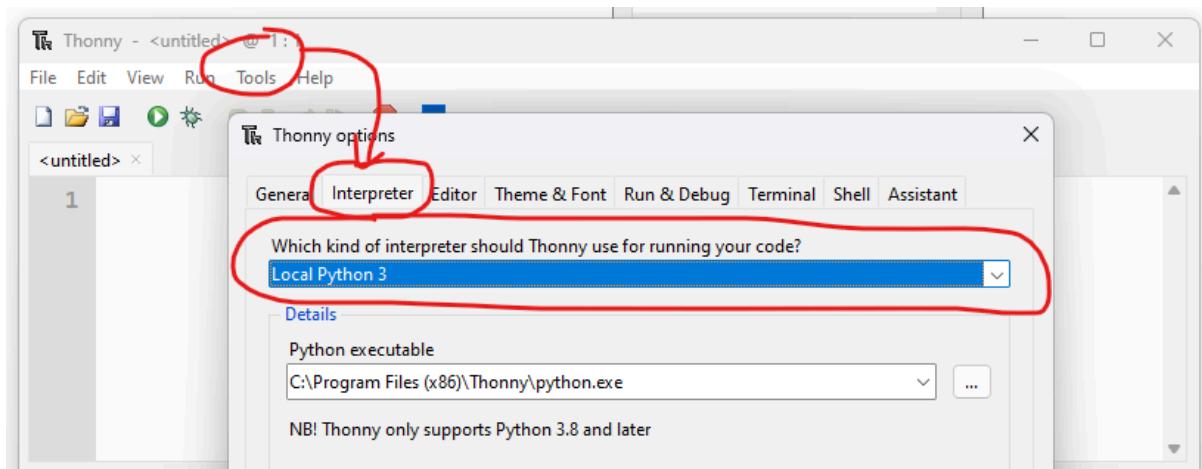
אפשרות א': התקנת MicroPython על בקר ESP32 תוך שימוש בסביבת הפיתוח Thonny

נוריד את הגרסה האחרונה של MicroPython מהאתר הבא:

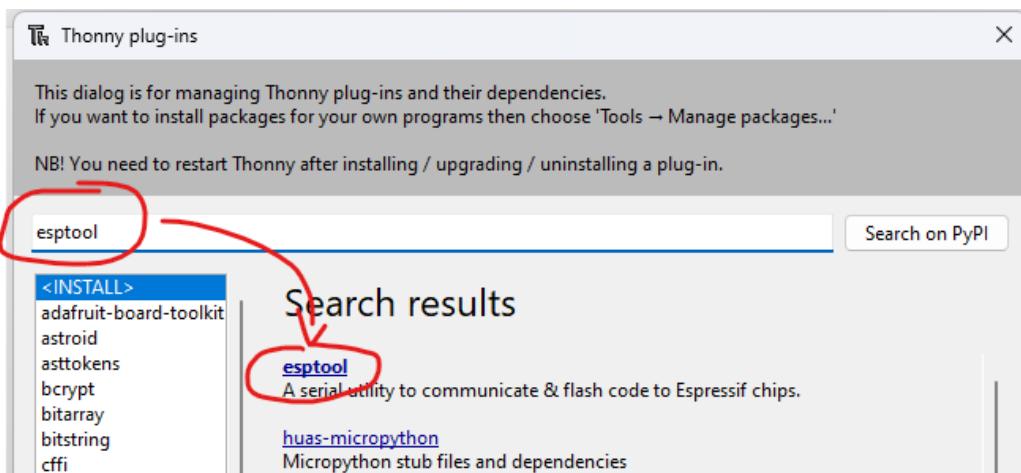
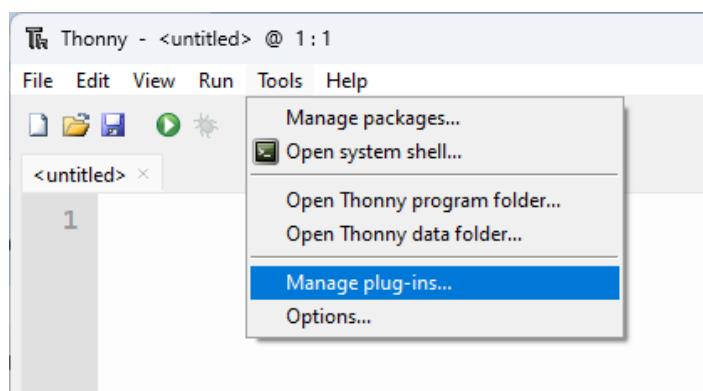
https://docs.espressif.com/projects/esp-at/en/latest/esp32/AT_Binary_Lists/esp_at_binaries.html

The screenshot shows the "ESP-AT User Guide" website. On the left sidebar, there are dropdown menus for "ESP32" and "master (latest)". Below them is a search bar and a "Get Started" button. Under "AT Binary Lists", there is a section for "ESP32 AT Released Firmware" which lists several series: ESP32-WROOM-32 Series, ESP32-MINI-1 Series, ESP32-WROVER-32 Series, ESP32-PICO Series, and ESP32-SOLO Series. There is also a link to "Subscribe to AT Releases" and "Brief Introduction to AT Firmware". The main content area is titled "Released Firmware" and contains a note in Chinese. It states that it is recommended to use the latest version of firmware. Currently, Espressif releases AT firmware for the following ESP32 series of modules. A "Note" section provides instructions for generating new firmware if no released firmware is available for a specific module. It lists four options: Modify UART Configuration, Modify Wi-Fi Configuration, Modify Certificate and Key Configuration, and Modify GATT Configuration. Below this is a section titled "ESP32-WROOM-32 Series" which lists three firmware versions: v3.4.0.0 ESP32-WROOM-32-AT-V3.4.0.0.zip (Recommended), v3.2.0.0 ESP32-WROOM-32-AT-V3.2.0.0.zip, and v2.4.0.0 ESP32-WROOM-32-AT-V2.4.0.0.zip. The first item is circled in red.

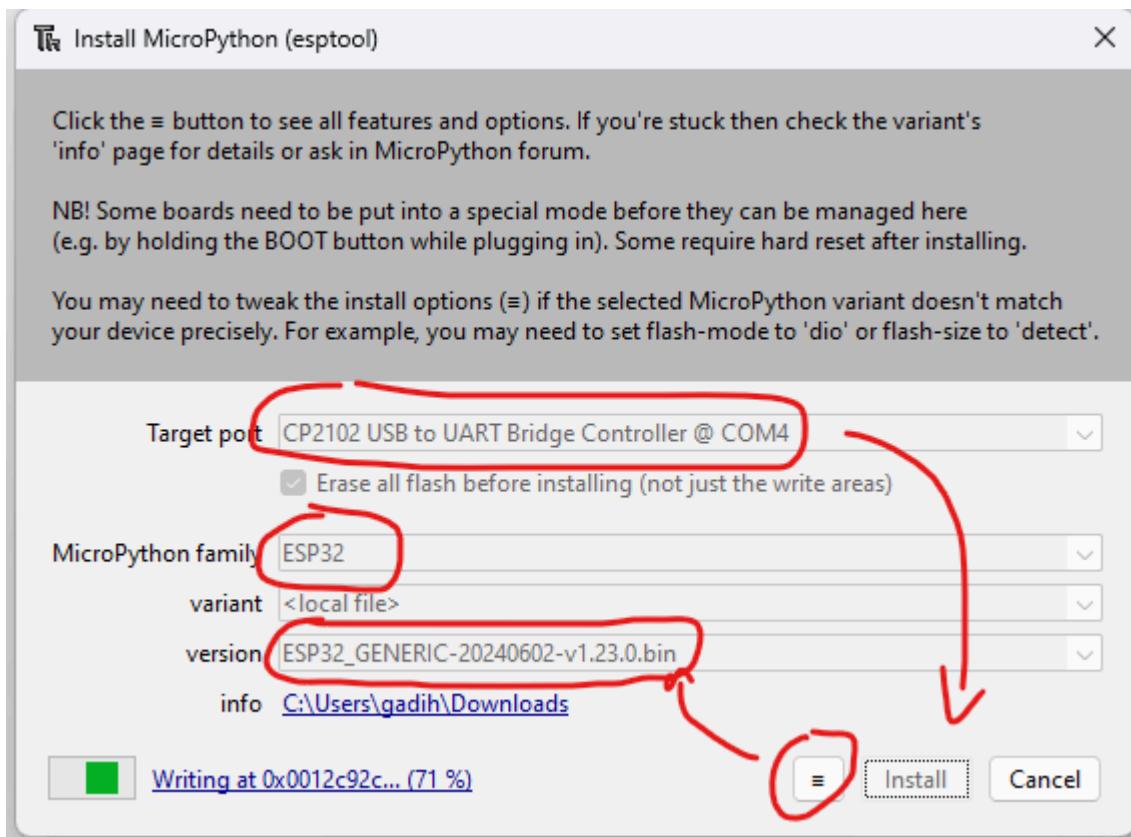
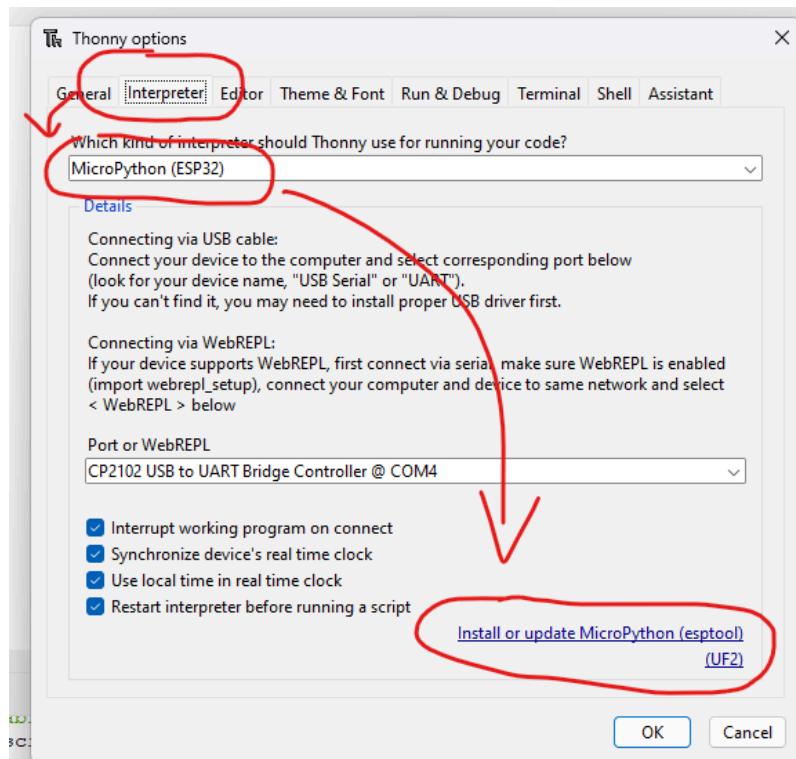
תחת התפריט Tools → Options → נחליף את שפת התוכנות מ- Python ל- MicroPython כה:



בutor סביבת העבודה נתקין את התוסף esptool.py המאפשר לנו לזרוב את הקושחה בברא:



נחזיר לתפריט Tools-> Options על הבקר: תחת לשונית את ה- MicroPythonInterpreter נתקין את ה- Tools-> Options על הבקר:



The screenshot shows the Thonny IDE interface. At the top, it says "Thonny - <untitled> @ 1:1". Below the menu bar is a toolbar with icons for file operations, run, stop, and other tools. The main area has a tab labeled "<untitled>". In the bottom left, there's a "Shell" tab. The shell window contains the following text:

```

Process ended with exit code None.

MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with ESP32
Type "help()" for more information.

MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with ESP32
Type "help()" for more information.

>>> |

```

A red circle highlights the second "MicroPython" line, and a red checkmark is placed to its right. At the bottom of the shell window, it says "MicroPython (ESP32) • CP2102 USB to UART Bridge Controller @ COM4".

זהו... סימנו להתקין גם את סביבת העבודה וגם את הקושחה המאפשרת תכונות בסביבת MicroPython על גבי בקר!!!ESP32

אפשרות ב': התקנת esptool.py על בקר ESP32 תוך שימוש ב- esptool.py

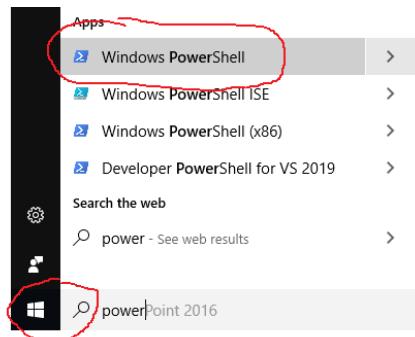
התקנת esptool.py

הכללי esptool.py מאפשר לנו לצרוב את מערכת הפעלה על הבקר.

מקור:

<https://github.com/espressif/esptool>

לשם ביצוע המשימה נפתח את התוכנה cmd או Windows PowerShell



```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\gadih> pip --version
pip 23.1.2 from C:\Users\gadih\AppData\Local\Programs\Python\Python311\Lib\site-packages\p
ip (python 3.11)
PS C:\Users\gadih> \|

```

נעוץ ב- **דיק להתקין את התוכנה אך לפני נבדוק שה- דיק מותקן על ידי היקלחת ההוראה הבא**

```
pip --version
```

דוגמיה לפולט תקין:

```

PS C:\Users\gadih> pip --version
pip 23.1.2 from C:\Users\gadih\AppData\Local\Programs\Python\Python311\Lib\site-packages\p
ip (python 3.11)

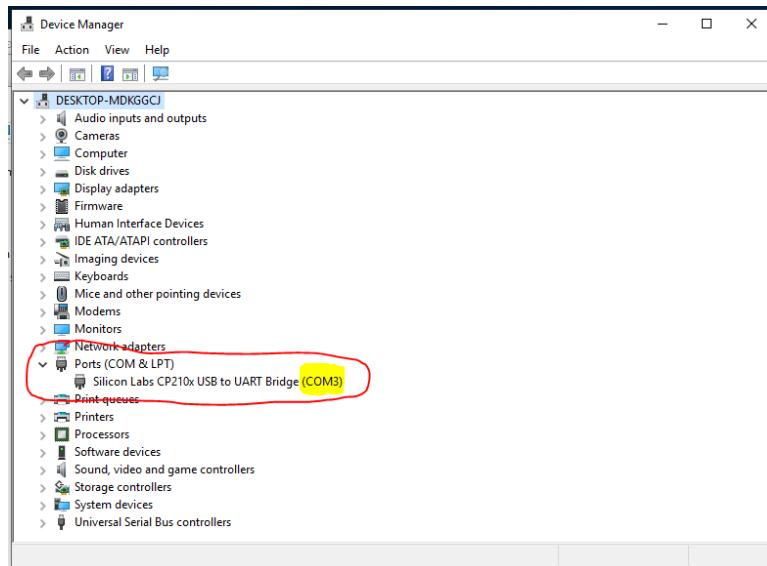
```

נעוץ ב- **דיק CDI להתקין את esptool**

```
> python -m pip install -U pip

> python -m pip install -U esptool
```

לאחר התקינה נחבר את הבקר ESP32 למחשב PC המריץ Windows 11 ונבדוק לאיזה com (מפתח מחשב) התחבר הבקר:



מערכת הפעלה איתרה את הבקר ב- COM3.

מקור:

https://micropython.org/download/ESP32_GENERIC/

בוצע אתחול של הבקר על ידי הוראה הבא

```
esptool --chip esp32 --port com3 erase_flash
```

נקבל לבקרה את הפלט הבא:

```
esptool.py v4.7.0
Serial port com4
Connecting...
Failed to get PID of a device on com4, using standard reset sequence.
.....
Chip is ESP32-D0WDQ6 (revision v1.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 24:0a:c4:c5:f4:10
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 8.6s
Hard resetting via RTS pin...
PS C:\Users\gadih>
```

***שים לב** לכך שצריך ללחוץ על לחץ BOOT בబקר כאשר המחשב מציג על המסך פלט של נקודות זו אחריו זו.

כדי לזרוב בబקר את מערכת הפעלה נוריד תחילה למחשב שלנו את הקובץ הבא:

https://micropython.org/download/ESP32_GENERIC/

Firmware

Releases

- v1.23.0 (2024-06-02) .bin / [.app-bin] / [.elf] / [.map] / [Release notes] (latest)
- v1.22.2 (2024-02-22) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
- v1.22.1 (2024-01-05) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
- v1.22.0 (2023-12-27) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
- v1.21.0 (2023-10-05) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
- v1.20.0 (2023-04-26) .bin / [.elf] / [.map] / [Release notes]
- v1.19.1 (2022-06-18) .bin / [.elf] / [.map] / [Release notes]
- v1.18 (2022-01-17) .bin / [.elf] / [.map] / [Release notes]
- v1.17 (2021-09-02) .bin / [.elf] / [.map] / [Release notes]
- v1.16 (2021-06-23) .bin / [.elf] / [.map] / [Release notes]
- v1.15 (2021-04-18) .bin / [.elf] / [.map] / [Release notes]
- v1.14 (2021-02-02) .bin / [.elf] / [.map] / [Release notes]
- v1.13 (2020-09-02) .bin / [.elf] / [.map] / [Release notes]
- v1.12 (2019-12-20) .bin / [.elf] / [.map] / [Release notes]

קחו בחשבון שיש למקם את הקובץ שהורד מהאתר בתיק'יה שבה ממוקה חלון ה - Windows PowerShell
במחשב שלך הורדתי את הקובץ לתיק'יה:

C:\Users\gadi>

ניתן לבדוק זאת על ידי הוראה זו כר:

```

PS C:\Users\gadih>
PS C:\Users\gadih> cd .\Downloads\
PS C:\Users\gadih\Downloads> ls ESP*
Red arrow points from the command 'ls ESP*' to the output table.

Directory: C:\Users\gadih\Downloads

Mode                LastWriteTime         Length Name
----                -----        ----  -
-a----       6/28/2024   9:38 AM      26404705 ESP32-WROOM-32-AT-V3.4.0.0 (1).zip
-a----       6/14/2024  10:57 AM      26404705 ESP32-WROOM-32-AT-V3.4.0.0.zip
-a----       6/29/2024   6:10 AM      1734240  ESP32_GENERIC-20240602-v1.23.0 (1).bin
-a----       6/18/2024  6:20 PM      1734240  ESP32_GENERIC-20240602-v1.23.0.bin

```

לביצוע הזריבה נשתמש בהוראה הבאה:

```

esptool --chip esp32 --port com3 --baud 460800 write_flash -z 0x1000
.\ESP32_GENERIC-20240602-v1.23.0.bin

```

נקבל לבקשתו את הפלט הבא:

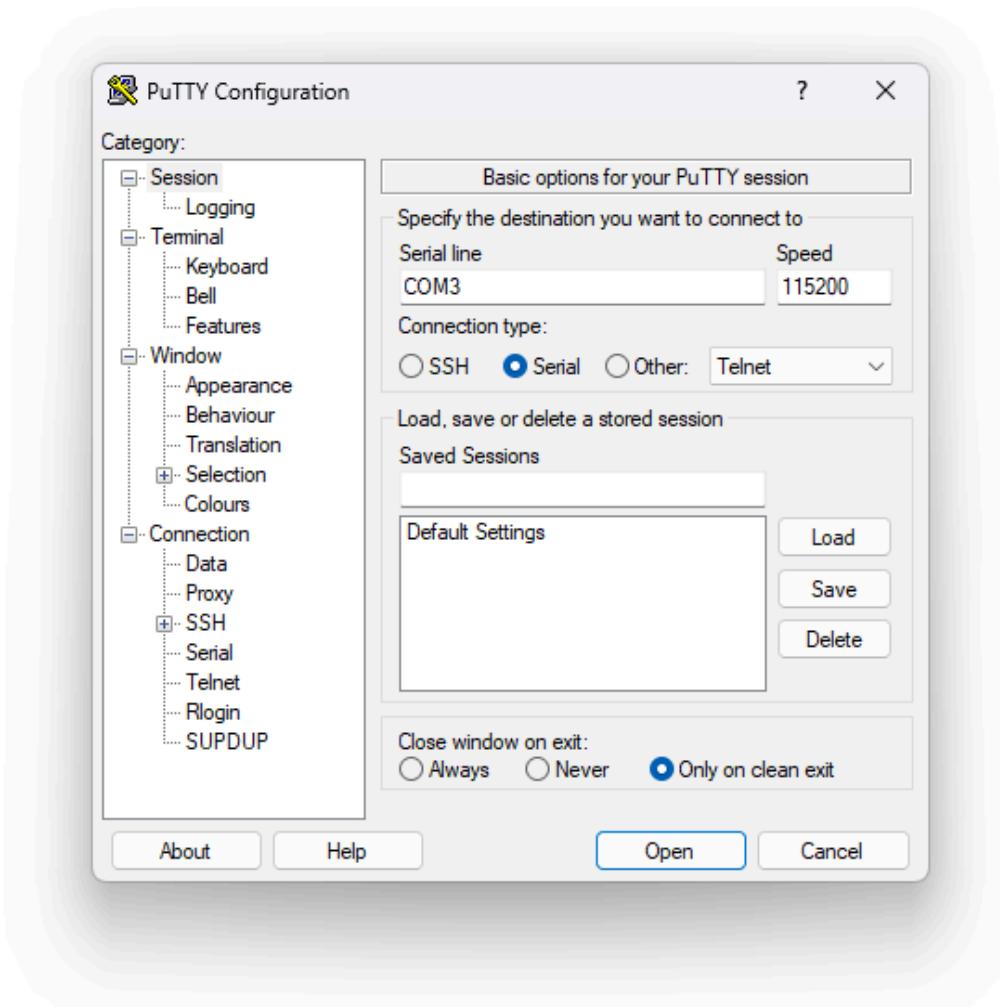
```

esptool.py v4.7.0
Serial port com4
Connecting...
Failed to get PID of a device on com4, using standard reset sequence.
.....
Chip is ESP32-D0WDQ6 (revision v1.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 24:0a:c4:c5:f4:10
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00001000 to 0x001a8fff...
Compressed 1734240 bytes to 1142447...
Wrote 1734240 bytes (1142447 compressed) at 0x00001000 in 26.5 seconds (effective 524.2 kbit/s)...
Hash of data verified.

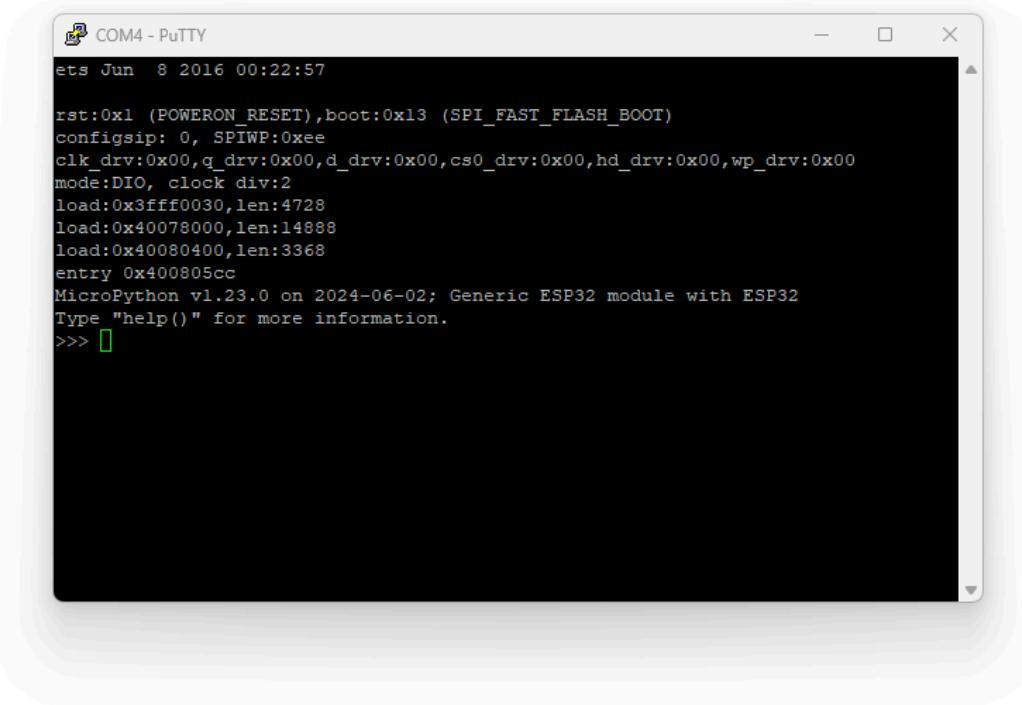
```

כדי לבדוק שהקוד נצרב בהצלחה ניתן להיתחבר לבקר דרך מפתח המחשב COM תוך שימוש בתוכנה כמו putty

[Download PuTTY: latest release \(0.81\) \(greenend.org.uk\)](https://www.greenend.org.uk/)



בהתחרות מוצלחת נקבל את הפלט הבא:



מכאן אנו יכולים לכתוב קוד ראשון ב-MicroPython לדוגמה:

```
>>> str1="Hello"
>>> str2="World"
>>> print(str1,str2)
Hello World
>>>
```

בשלב זה ניתן גם לכתוב קוד בסיס להדלקה וכיובי נורות לד. לדוגמה:

```
>>> from machine import Pin
>>> led=Pin(2,Pin.OUT)
>>> led.value(1)
>>> led.value(0)
>>>
```

משימה 2 - כתיבת קוד MicroPython בסביבת הפיתוח Visual Studio Code

מקור:

<https://docs.pycom.io/gettingstarted/software/vscode/>

<https://www.youtube.com/watch?v=YOeV14SESIs>

אם אתם מתקננים מנוסים שמכירים את סביבת הפיתוח Visual Studio, פועלות זו מיועדת לכם. אחרת המשיכו לכתוב תוכנה תוך שימוש בסביבת הפיתוח Uhoh! כפי שלמדנו בפעולות הראשונה.

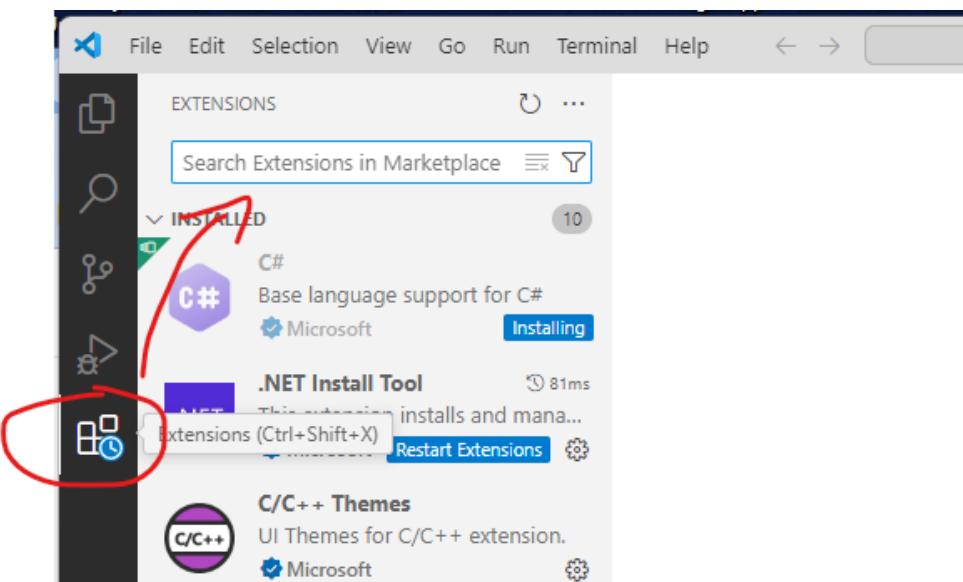
בכותרת Visual Studio Code הוא עורך קוד חינמי, קל משקל הנitin להרחבה לבניית אפליקציות ולתיקון בעיות. הוא פועל על מערכות הפעלה Windows, macOS ו-Linux. סביה זו מאפשרת להתקין לה תוספים המאפשרים לה גמישות גדולה בכל הקשור לשפות התכנות והפלטפורמות שהיא תומכת בהם.

בפעולות זו נתקין את התוסף PyMakr המאפשר לנו לכתוב קוד ב-MicroPython לבקרי ESP32.

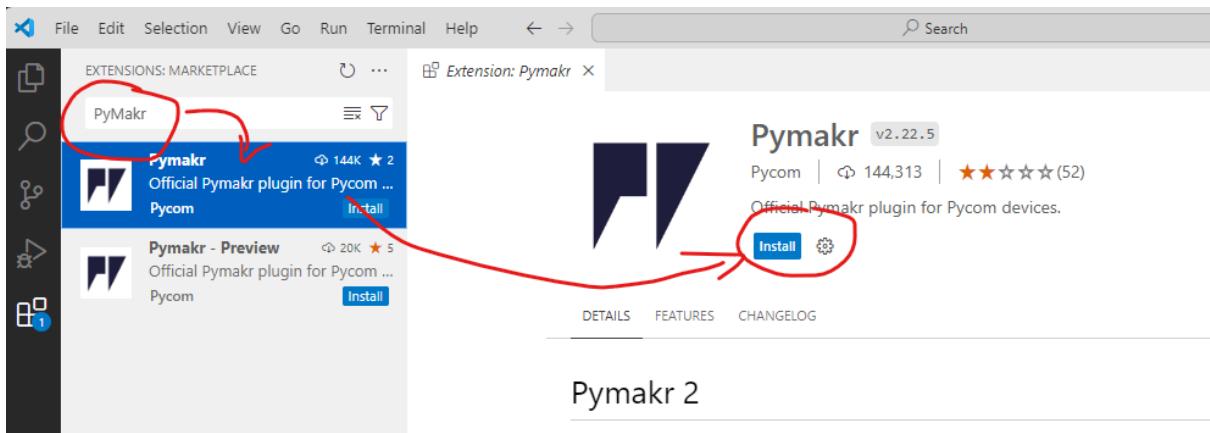
בסוף זה דרוש התקינה של SJNode על המחשב המקומי שעליו אתם עובדים. ניתן להוריד את התוסף מהאתר:

<https://nodejs.org/>

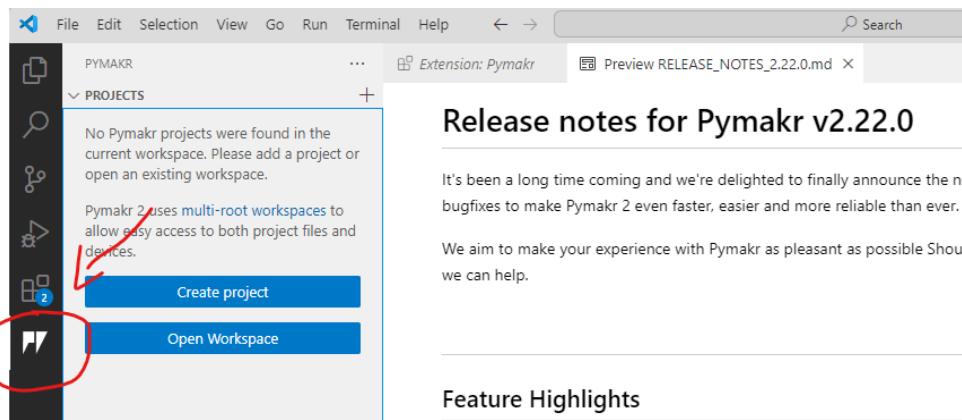
נפתח את סביבת הפיתוח Visual Studio Code ונלחץ על **תוספים** (Extensions)



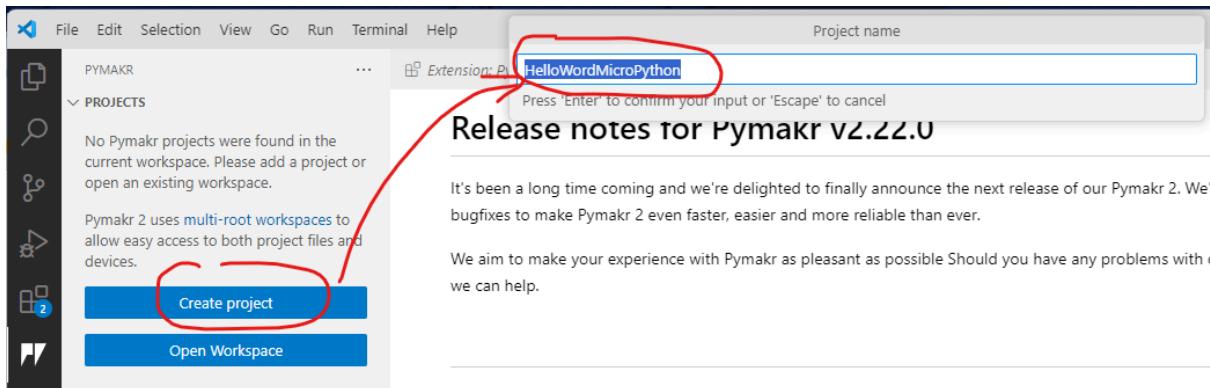
נחפש את התוסף :PyMakr



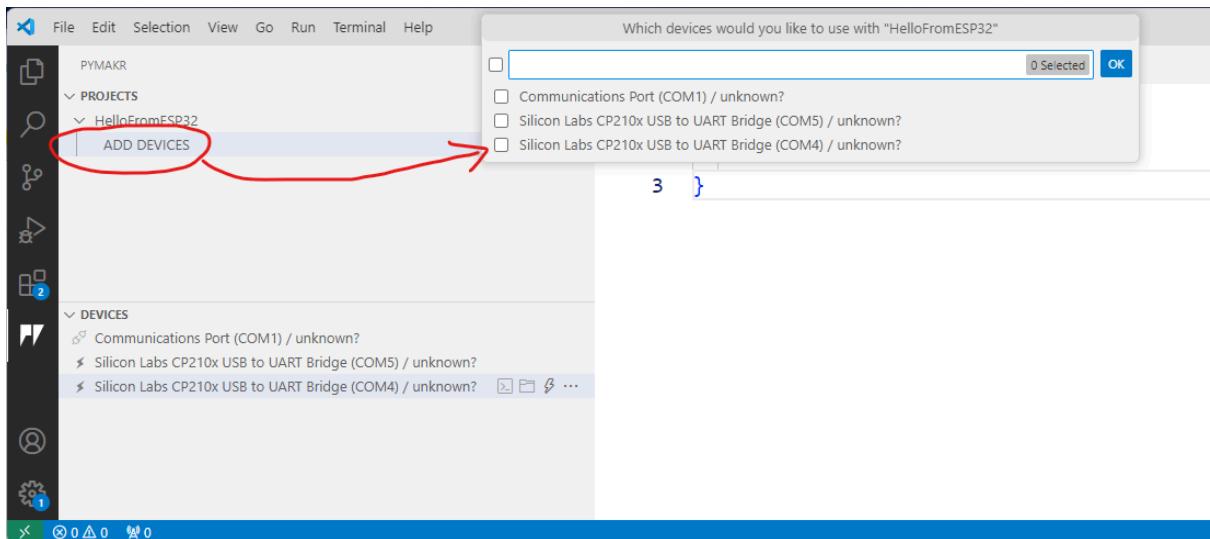
לאחר ההתקינה נקלט לחץ חדש בסביבת העבודה:



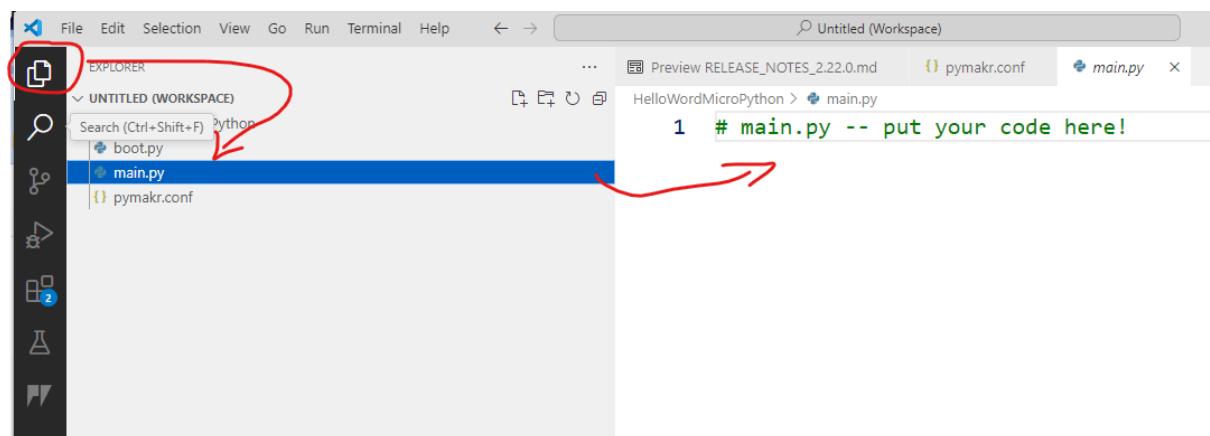
לחץ על הלחץ :Create project



נקבל את החלון הבא וنبחר בו את הרכיב שアイתו נעובד:



לאחר בחירת הרכיב נעבור לחלון הפROYIKט:

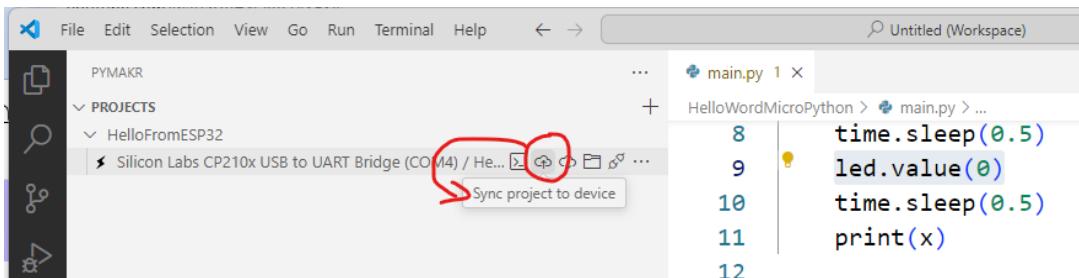


נפתח את קובץ הקוד `main.py` ונכתב בו את הקוד הבא:

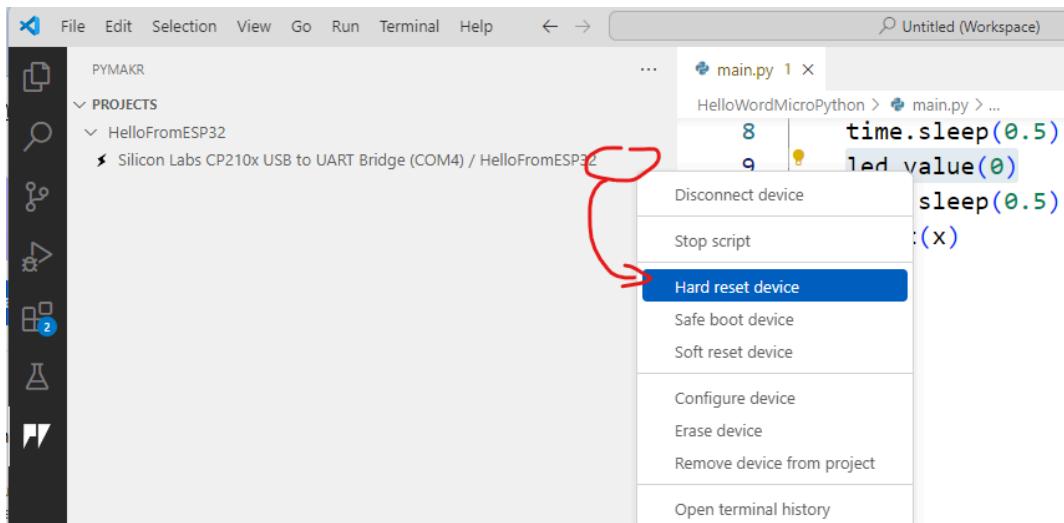
```
import time
from machine import Pin
led=Pin(32,Pin.OUT)

for x in range(10):
    led.value(1)
    time.sleep(0.5)
    led.value(0)
    time.sleep(0.5)
print(x)
```

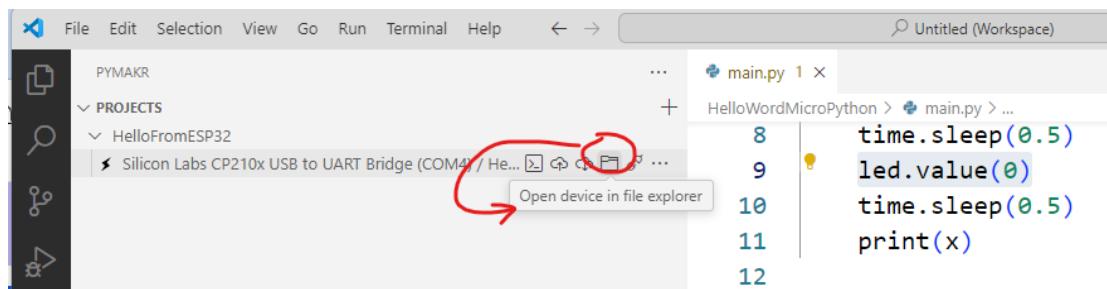
כדי להעביר את התוכנה שכתבנו לבקר נלחץ על הלץ הבא:



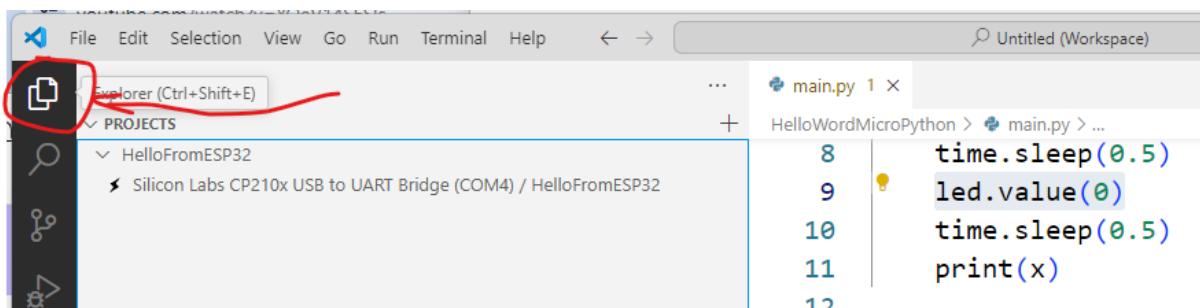
כדי להריץ את הקוד בברker נלחץ על:



ניתן לתקן ולעורך קוד ישירות על גבי הAKER. כדי לעשות זאת יש לפתח את הפרויקט ששמור בAKER בחלון ה-Explorer על ידי לחיצה על "Open device in file explorer":



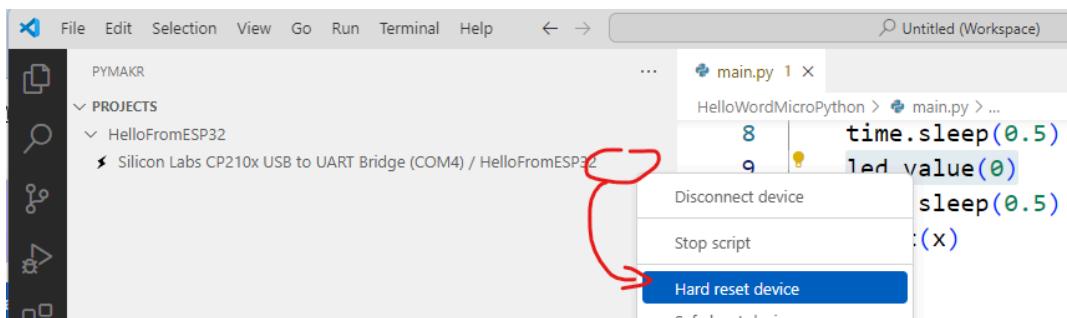
ולאחר מכן נלחץ על לחץ על ה-Explorer:



```
# main.py -- put your code here!
import time
from machine import Pin
led=Pin(32,Pin.OUT)

for x in range(10):
    led.value(1)
    time.sleep(2)
    led.value(0)
    time.sleep(1)
print(x)
```

בשלב זה ניתן לתקן ולערוך את הקוד שבבקיר וצחריץ אותו על ידי:



משימה 3 - כתיבת תוכנית ראשונה לביצוע פלט בשפה Python

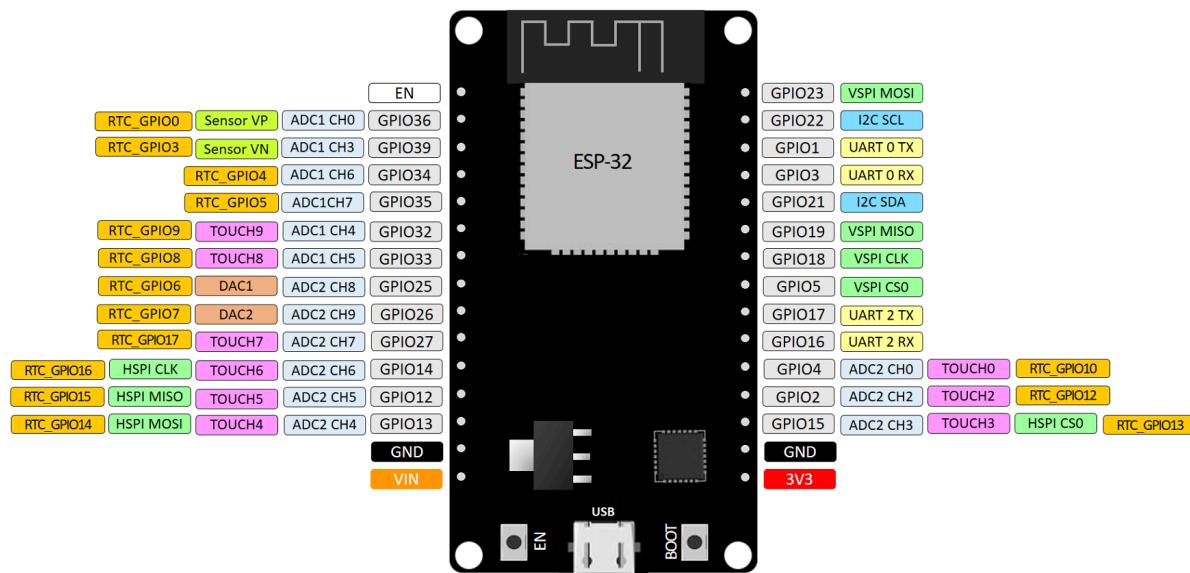
מקור:

<https://myhomethings.eu/en/esp32-pinout-which-pin-is-for-what/>

<https://electropeak.com/learn/full-guide-to-esp32-pinout-reference-what-gpio-pins-should-we-use/>

בניסוי זה נתרgal כתיבת תוכנית ראשונה בפייטון העשוה שימוש בפעולות פלט למפתח הבקר.

נחבר למחשב את הבקר ESP32 דרך מפתח ה- USB



*קיים בשוק מוצר ESP32 שבמהלכו הרכיב ממופים באופן שונה מהמתואר כאן.

מהאויר ניתן ללמוד שכרטיס הפעיתו המתואר מספק לנו גישה ל- 25 דקקי GPIO. כמו כן ניתן לראות שככל אחד מהדקקי ה- GPIO מספק מספר יישומים נוספים כמו מבואות אנלוגיים, הדקי תקשורת מסווגים שונים ומוצאים אנלוגיים.

שליטה בהדקקי ה- GPIO של בקר ה- ESP32 נעשית על ידי שימוש במחלקה Pin המהווה חלק מהמספרייה machine.

נדרש להגדיר עצם מהמספרייה Pin עבור כל הדק GPIO שנרצה להפעיל.

הקוד הבא מגדים 2 עצמים led1 ו-led2 המהווים מופעים של המחלקה Pin מופעים אלה ממופים להדקים 25 ו-32 של הבקר ומגדירים כמוצאים.

```
from machine import Pin

led2=Pin(25,Pin.OUT)
```

ניתן להגדיר כבר בשלב האתחול של העמם את מצבו הלוגי. לדוגמה:

```
pinLed = Pin(32, mode=Pin.OUT, value=1) # 3.3V on output
time.sleep(2)
```

```
pinLed = Pin(32, mode=Pin.OUT, value=0) # 0V on output
```

המחלקה Pin מספקת לנו מספר פעולות לשינוי מצבו הלוגי של הדק. נדגים זאת:

```
LedPin2 = Pin(2, mode=Pin.OUT)

LedPin2.on()      # Set 3.3V at the output (high logic state)
LedPin2.high()    # Set 3.3V at the output (high logic state)
LedPin2.value(1)   # Set 3.3V at the output (high logic state)

time.sleep(2)

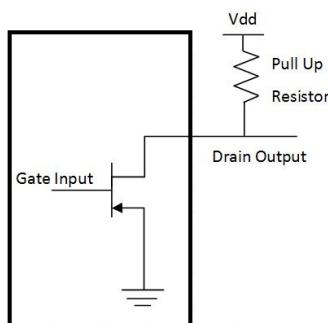
LedPin2.off()     # Set 0V at the output (low logic state)
LedPin2.low()     # Set 0V at the output (low logic state)
LedPin2.value(0)   # Set 0V at the output (low logic state)
```

חיבור התקני קלט להדק - GPIO

ראינו שאשר אנו רוצים לעבוד עם אחד מהדקי פלט של רכיב, אנו נדרשים לאתחל עצם מהמחלקה Pin. באופן דומה כאשר אנו רוצים להגדיר הדק קלט נכתב את הקוד הבא:

```
from machine import Pin
led=Pin(15,Pin.IN)
```

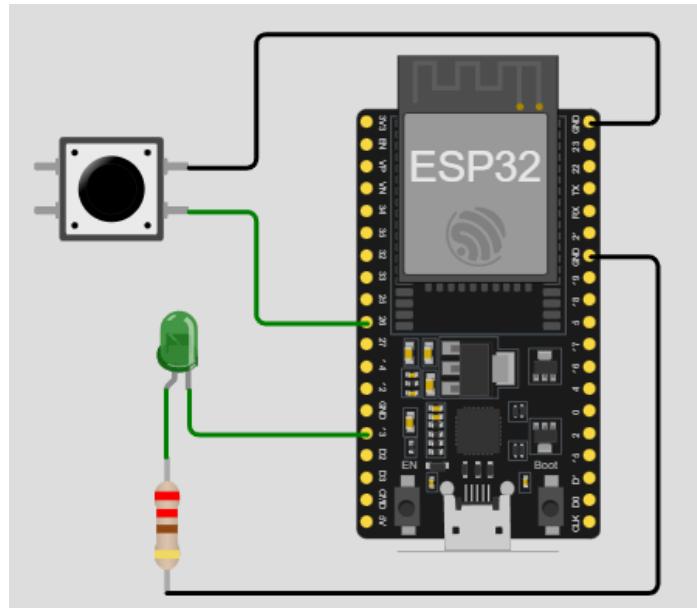
ניתן לאתחל את העצם המיצג את ההדק בדרכים נוספות כמו הגדרת ההדק כ- open-drain output (מצב שבן נקלט עבר 1 לוגי הדק מוצא מקוצר לאדמה וב- 0 לוגי נקלט נתוך).



כמו כן ניתן לאתחל את ההדק באמצעות פתרים נוספים כmodo: Pin.PULL_UP או Pin.PULL_DOWN פתרים נוספים ניתן בקישור הבא:

<http://docs.micropython.org/en/v1.11/library/machine.Pin.html#machine-pin>

להלן שימוש בסיסי להדלקת נורת LED על ידי מפסוק:



להלן קוד התוכנית:

```
from machine import Pin

pin_button = Pin(26, mode=Pin.IN, pull=Pin.PULL_UP)
pin_led = Pin(13, mode=Pin.OUT)

pin_led.on()

while True:
    if pin_button.value() == 1:
        pin_led.on()
    else:
        pin_led.off()
```

תרגום:

כמה פעמים ידליק ה- LED כאשר נורץ את הקוד הבא בברך:

```
from machine import Pin
import time

pin_led = Pin(13, mode=Pin.OUT)
```

```

pin_led.on()
time.sleep(0.5)
pin_led.off()
time.sleep(0.5)

for x in range(1,10,2):
    pin_led.value(1)
    time.sleep(0.5)
    pin_led.value(0)
    time.sleep(0.5)
print(x)

```

תשובה:

6 פעמים

תרגום:

כתב תוכנית לマイקו בקר ממשפחת EPS32 בשפת MicroPython המבצע את הפעולות הבאות:

1. מגדר את הדק 15 של הבקר בפתח מוצא.
2. נורית LED המתחברת לדק 15 תהבהב 10 שניות בקצב של 100 מייל. שניות (0.1 שניות)
3. בהמשך תדלק הנורית למשך 3 שניות נוספת ואז תכבה.

תשובה:

```

import time
from machine import Pin
led=Pin(15,Pin.OUT)

for x in range(100):
    led.value(1)
    time.sleep(0.1)
    led.value(0)
    time.sleep(0.1)

    led.value(1)
    time.sleep(3)
    led.value(0)

```

שילוב השהיות בתוכנה.

ניתן לשלב הוראות השהייה בשלוש טווחי זמן שונים: שניית, אלףיות השניה, מיליוןיות השניה להלן דוגמה:

```
import time

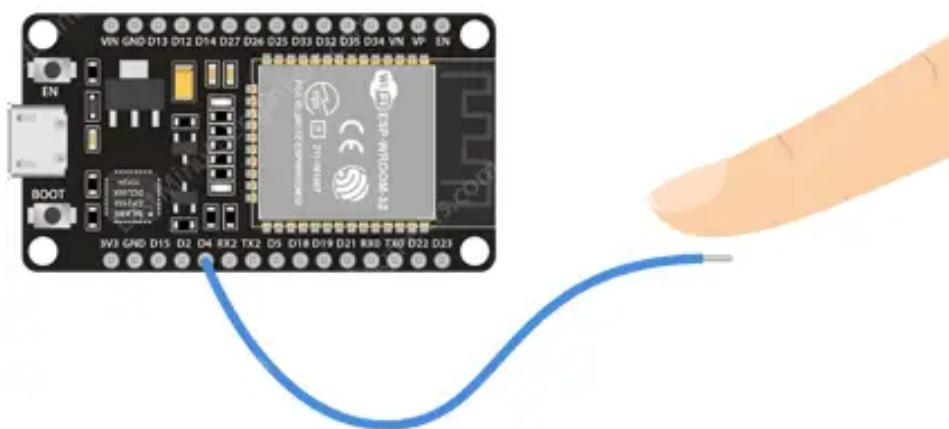
time.sleep(1)          # sleep for 1 second
time.sleep_ms(500)     # sleep for 500 milliseconds
time.sleep_us(10)      # sleep for 10 microseconds
start = time.ticks_ms() # get millisecond counter
delta = time.ticks_diff(time.ticks_ms(), start) # compute time difference
```

שימוש בחישבי מגע קיבוליים מובנים ב-ESP32

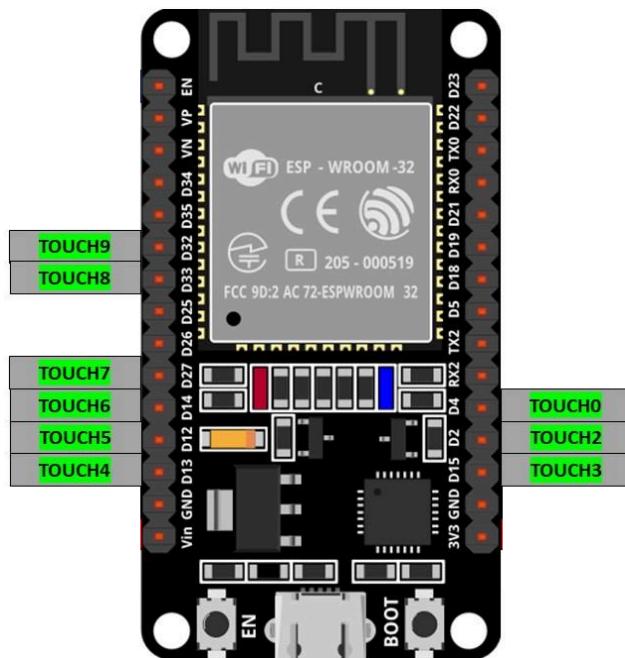
בקר ה-ESP32 מכיל חישבי מגע קיבוליים מובנים (Capacitive Touch Sensors) אשר יכולים לזהות מגע אנושי ללא צורך בחומרה נוספת. חישבים אלו רגישים לשינויים בקיול ומאפשרים יצירת משקע משתמש אינטואיטיביים.

עקרון הפעולה של דק' Touch:

חישבים קיבוליים ב-ESP32 פועלים על ידי מדידת שינוי הקיבול החשמלי של הדק. כאשר האצבע של לנו **מתקרבת** או נוגעת בדק, מתרחש שינוי בקיול עקב התכונות החשמליות של גוף האדם. ה-ESP32 מודד את הזמן שהוקח להדק להיטען או להתפרק, מדידה זו משתנה בהתאם לקיבול שנוצר בין הדק לבין הגוף שלנו. הערך המוחזר הוא מספר המציין את זמן הפעינה/פריקה - ערך נמוך יותר מאשר מגע.



ל-ESP32 יש מספר דק' מגע קיבוליים (T0, T1, T2, T3, T4, T5, T6, T7, T8, T9). לא כל הדקים זמינים בכל לוחות ESP32, יש לבדוק את מסמכי הלוח הספציפי.



להלן דוגמאות קוד בסיסית להפעלת הדק Touch:

```
from machine import TouchPad, Pin
import time

# Initialize the touch pin (GPIO 4)
# The touch pin is usually a capacitive touch sensor pin
touch_pin = TouchPad(Pin(4))

while True:
    touch_value = touch_pin.read()
    print("touch_value:", touch_value)
    time.sleep(0.5)
```

דוגמה לדיזהו מגע עם סף קבוע:

```
from machine import TouchPad, Pin
import time

# Initialize the touch pin (GPIO 4)
touch_pin = TouchPad(Pin(4))

# Adjust this threshold based on your touch sensor's sensitivity
TOUCH_THRESHOLD = 450

while True:
    touch_value = touch_pin.read()
    if touch_value < TOUCH_THRESHOLD:
        print("מגע מזוהה!")
    else:
        print("אין מגע")
    time.sleep(0.1)
```

דוגמה לכיוון אוטומטי של סף המגע:

```
from machine import TouchPad, Pin
import time

touch_pin = TouchPad(Pin(4))

print("בצע כיוול - אל תיגע בדק")
time.sleep(2)
samples = 10
sumsamples = 0
baseline = 0
for _ in range(samples):
    sumsamples += touch_pin.read()
baseline = (sumsamples / samples)
THRESHOLD = baseline * 0.7 # 30% ירידת מערך הבסיס
print(f"ערך בסיס: {baseline}, סף: {THRESHOLD}")

while True:
    touch_value = touch_pin.read()
    if touch_value < THRESHOLD:
        print("!מגע מזוהה!")
    time.sleep(0.1)
```

דוגמה לשימוש במגע להדלקת נורת LED:

```
from machine import TouchPad, Pin
import time

touch_pin = TouchPad(Pin(4))
led = Pin(2, Pin.OUT) # LED on GPIO 2

print("בצע כיוול - אל תיגע בדק")
time.sleep(2)
samples = 10
sumsamples = 0
baseline = 0
for _ in range(samples):
    sumsamples += touch_pin.read()
baseline = (sumsamples / samples)
THRESHOLD = baseline * 0.7 # 30% ירידת מערך הבסיס
print(f"ערך בסיס: {baseline}, סף: {THRESHOLD}")

while True:
    touch_value = touch_pin.read()
    if touch_value < THRESHOLD:
        led.on()
    else:
        led.off()
    time.sleep(0.1)
```

טיפים ופתרון בעיות

1. רגישות יתרה: אם החישון מזזה מגע ללא סיבה, יש להגדיל את ערך הסף.
2. חוסר רגישות: אם החישון לא מזזה מגע, יש להקטין את ערך הסף או לבדוק את החיווט.
3. הפרעות אלקטרומגנטיות: מקורות חשמל חזקים יכולים להשפיע על הקראות.

לסיכום, דקי המגע הקיבוליים ב-ESP32 מספקים דרך פשוטה ויעילה להוספה ממשך מגע לפרויקטים. באמצעות MicroPython, ניתן לקרוא את ערכי החישונים ולהציג למגע בקלות. חשוב לציין לביצוע כיוול מתאים ולבחר ערכי סף נכונים עבור כל יישום ספציפי.

משימה 4 - שימוש ב-Timer פונימי

כמו בכל בקר גם ב- ESP32 קיימ Timer להלן דוגמה לעובדה עם הרכיב.

קישורים:

https://www.youtube.com/watch?time_continue=42&v=Mku1Bq78nXw&feature=emb_logo

<http://docs.micropython.org/en/v1.11/library/machine.Timer.html#machine-timer>

ESP32 SoC	Number of timers
ESP32	4
ESP32-S2	4
ESP32-S3	4
ESP32-C3	2
ESP32-C6	2
ESP32-H2	2

בקוד הבא נגרום לנורית LED המתחוברת להדק 13 של הבקר להבהב בקצב של שנייה

```
from machine import Pin, Timer

led1 = Pin(13,Pin.OUT)

def handleTimerInt(timer):
    led1.value(not led1.value())

myTimer = Timer(0)
myTimer.init(period=1000, mode=Timer.PERIODIC, callback=handleTimerInt)
```

חשוב: שימוש לב שהבהוב נורית ה- LED על ידי ה- Timer אינה גורמת לבקר להיתקע, לעומת זאת ניתן להמשיך להפעיל את הבקר במקביל לפעולות ה- Timer

ננצל את זה שהברר פוני ונציג על המסך את מספר הפעמים שהייתה פסיקת Timer:

```
from machine import Pin, Timer

interruptCounter = 0
```

```

led1 = Pin(13,Pin.OUT)

def handleTimerInt(timer):
    led1.value(not led1.value())
    global interruptCounter
    interruptCounter = interruptCounter+1
    print("Interrupt has occurred: " + str(interruptCounter))

myTimer = Timer(0)
myTimer.init(period=300, mode=Timer.PERIODIC, callback=handleTimerInt)

```

נקבל את הפלט הבא בנוסף לפעולות הבהוב של נורית ה-LED

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS Silicon Labs CP210x USB to UART Bridge (COM4) / Led Example

Interrupt has occurred: 172
Interrupt has occurred: 173
Interrupt has occurred: 174
Interrupt has occurred: 175
Interrupt has occurred: 176
Interrupt has occurred: 177
Interrupt has occurred: 178

```

שימוש ב-Watchdog Timer ב-ESP32

Watchdog Timer הוא מנגנון בטיחותי המיועד לאתחול מערכת באופן אוטומטי כאשר מתרחשת תקלה או כשר התוכנה "נתקעת" (לדוגמה, עקב לולאה אינסופית או חוסר תגובה). ה-WDT פועל כטימר ספירה לאחר מכן אם התוכנה אינה מתחילה מחדש את הטימר באופן קבוע, הוא ייגרום לאתחול המערכת כאשר הוא מסיים ספורה.

שימוש נפוץ ב-WDT כולל:

- מניעת השבתה של מכשירים מרוחקים עקב באגים בתוכנה.
- שחרור אוטומטי של מערכת לאחר קriseה.
- הגנה מפני תוכנה שאינה מגיבה עקב הדרישות חיצונית.

הגדרת Watchdog Timer

ב-*MicroPython*, ה-Watchdog Timer זמין דרך הספרייה `machine.WDT`. להלן דוגמה לאופן הייבוא שלו לתוכנית:

```
from machine import WDT
```

צירוף עצם מסווג כ-Watchdog Timer

ניתן להגדיר את זמן הספירה לאחר (באלפיות שנייה). אם לא יאותחל מחדש לפני תום הזמן, המערכת תתאפס.

```
# It initializes the WDT with a timeout of 5000 milliseconds (5 seconds).
wdt = WDT(timeout=5000)
```

אתחול מחדש של ה-Watchdog Timer

כדי למנוע אתחול, יש לקרוא לפעולה feed לפני תום זמן הזמן timeout. נדגים זאת:

```
wdt.feed() # This line resets the WDT timer.
```

דוגמה מעשית

תוכנית הכוללת לפחות ראשית, ה-WDT יופס בכל איטרציה כדי למנוע אתחול של הבקר.

```
from machine import WDT
import time

wdt = WDT(timeout=5000)

while True:
    print("Running...")
    # Uncomment the next line to see the watchdog timer in action
    wdt.feed() # Reset the watchdog timer
    time.sleep(1) # Simulate some work
```

לצורך ההמחשה ניתן למחוק את השורה:

```
wdt.feed() # Reset the watchdog timer
```

ולראות שהבקר מתאפשר לאחר 5 שניות.

דוגמה נוספת לשימוש ניתן לראות בקוד הבא שבו נוסף לולאה אינסופית פנימית, ה-WDT לא יאותחל מחדש והמערכת תתאפשר לאחר 5 שניות. נדגים זאת:

```
from machine import WDT
import time

wdt = WDT(timeout=5000)

while True:
    print("Running...")
    wdt.feed()
    time.sleep(1)

    סימולציה # לולאה אינסופית
    while True:
        pass
```

הקבצים boot.py ו main.py

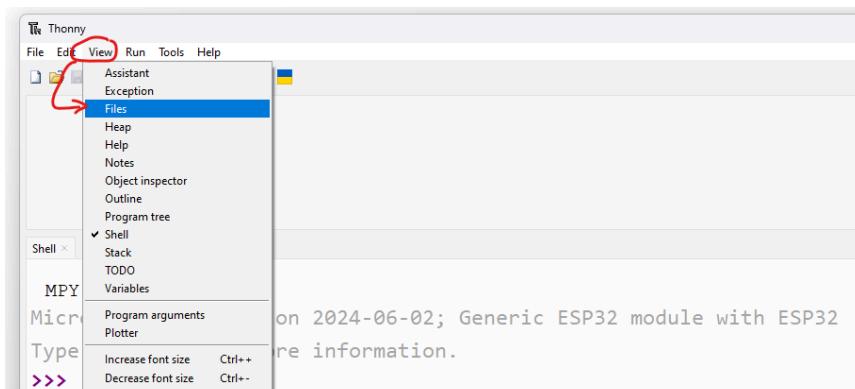
לכל בקר מוגדרים 2 קבצים ייחודיים הנשמרים בו. הראשון קובץ בשם boot.py והשני קובץ בשם uc.main. בקובץ boot.py משמש אותנו להקצת קוד עבור הוראות אתחול של הבקר, קוד זה רץ פעמיים אחד בלבד בזמן האתחול. בקובץ זה מקובל ליבא ספריות רלוונטיות, הגדרת קבועים כמו שמות ויסודות. במשימה זו נעשה שימוש בקובץ זה כדי ליצר לבקר את הקישורית לרשות האינטרנט.

הקובץ uc.main עתיד להכיל את קוד התוכנית שתעבד באופן אוטומטי לאחר הרמת הקובץ uc.boot קובץ זה ישימושו להפעלת היישום שכתבנו לאחר שלב הפיתוח.

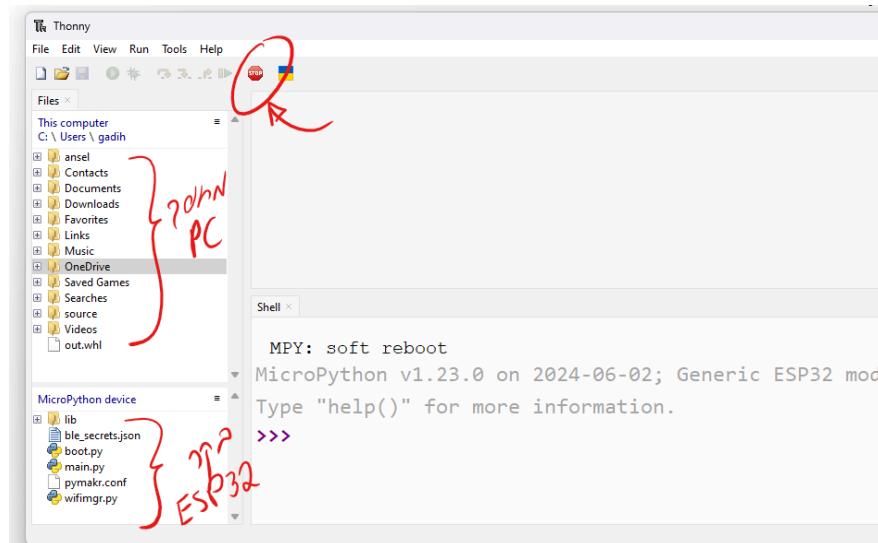
הערה: ברוב הפעמים כאשר כותבים קוד תוכנה עבור בקר בשפת MicroPython תוכלו למצאו שני קבצים המאוחסנים ב.tkernר עצמו: הקובץ הראשון נקרא: boot.py והשני uc.main. ברגע שה.tkernר מקבל מתח עבודה או מיד לאחר איפוס יוזם, הבקר מפעיל אוטומטית את הקוד השמור בקובץ uc.boot לאחר מכן הוא יפעיל את תוכן הקובץ uc.main ממשמעות הדבר היא שלאחר שמירית הקובץ uc.main ב.tkernר לא ניתן היה לבצע עדכוני תוכנה בו ללא מחיקת הקובץ הנ"ל כאיו מדבר באתחול בקר חדש. לנ' השימוש באפשרות זו רק כאשר אתם מעבירים את הקוד ממצב פיתוח למצב שימוש.

כתיבת הקובץ boot.py

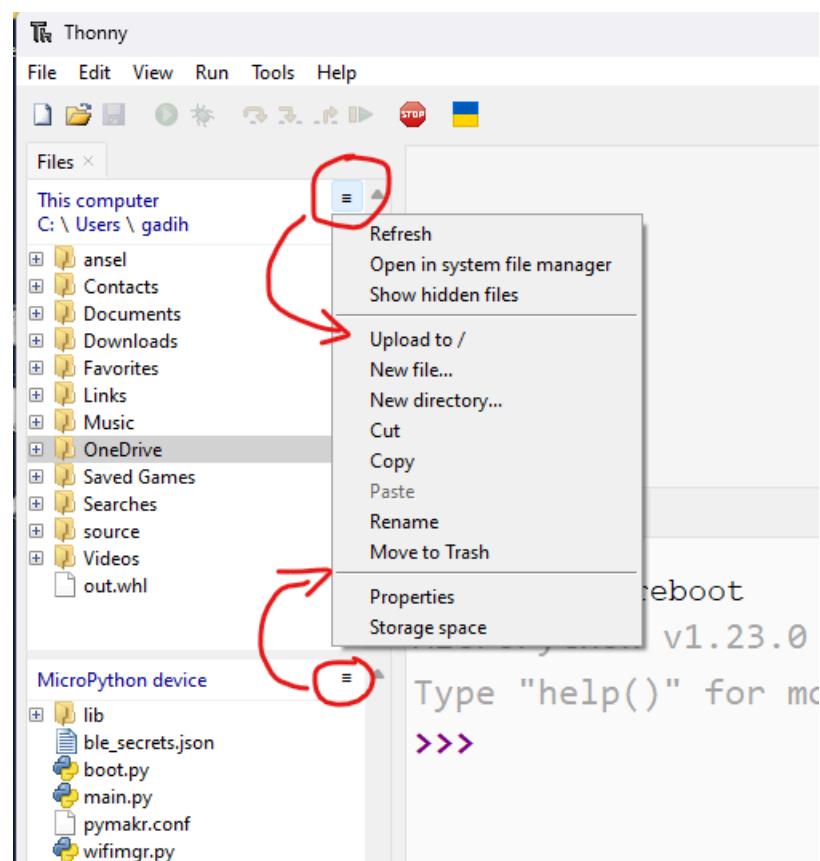
בקר ESP32 מצוייד בזיכרון הרצק בפח של 4 מגה-בייט שבו ניתן לשמר קבצי קוד ונתונים. נעזר בסביבת הפיתוח Thonny כדי לבדוק את הקבצים שבחיצון ה.tkernר מיד לאחר אתחול בקר בקורסיה חדשה. נעשה זאת על ידי לחיצה על View -> File כדי לראות את הקבצים שב.tkernר:



נפתח לנו המסר הבא:



בעזרתחלון שפתחנו אנו יכולים לראות אילו קבצים שמורים בזיכרון של הAKER. כמו כן ניתן לבצע את כל הפעולות הבסיסיות על הקבצים כמו הוספה קבצים, מחיקת קבצים, יצירה ומחיקה של תיקיות ושינוי של כל קובץ.



נעוז במכשיר שפתחנו כדי להעתיק את הקובץ `boot.py` מהזיכרון של הAKER למחשב על ידי לחיצה על Upload to

לטיכום: ה-Watchdog Timer הוא כלי חיוני בפיתוח פרויקטים משובצות מחשב. ניתן להשתמש בו-WDT כדי להגן על המערכת מפני תוכנה שאינה מגיבה. חשוב לתקן את נקודות האתחול (feed) בצורה חכמה כדי למנוע אתחולים לא רצויים אך עדים לאפשר שחזור במקרה של תקלת.

משימה 5 - פסיקות חומרה

פסיקות חומרה מאפשרות לבקר לטפל ביעילות בשינויים המתרחשים בחומרה. הטיפול בפסיקה אינו מחייב לבדוק כל הזמן את מצב החומרה אלה כאשר מתגלה שינוי, מופעל אירע הממומש כפונקציה בתוכנה. כאשר מתרחשת הפסיקה, המעבד מפסיק את ביצוע התוכנית הראשית, מבצע פונקציה מוכנה מראש ואז חוזר לתוכנית הראשית.

בבקר ESP32 ניתן להגדיר פסיקות לכל הדקי הבקר פרט להדקים GPIO6 ו-GPIO11.

קישורים:

<https://randomnerdtutorials.com/micropython-interrupts-esp32-esp8266/>

<https://docs.micropython.org/en/latest/library/machine.Pin.html>

<https://techtutorialsx.com/2017/10/08/esp32-micropython-external-interrupts/>

אחד היישומים למנגנון הפסיקות הוא שימוש פסיקה כדי לטפל בלחיצה על לחץ. נדגים זאת על ידי הקוד הבא:

```
from machine import Pin

def on_pressed(pin):
    print(pin)
    print('pressed')

# Setup the button input pin with a pull-up resistor.
button = Pin(33, Pin.IN, Pin.PULL_UP)

# Register an interrupt on rising button input.
button.irq(on_pressed, Pin.IRQ_RISING)
```

בקוד זה אנו מגדירים עצם בשם button הקולט לחץ שנמחובר לדק 33 של הבקר וכלל נגד PULL_UP הממומש בתוך הרכיב.

ההוראה button.irq מגדירה להדק זה פיסקה שתתפעל בעבר מנמוך לגבוה (IRQ_RISING). ברגע שהפסיקה תתקבל נזון את הפעולה on_pressed בו למבצע קוד הפסיקה.

ניתן להשתמש במאפיין trigger כדי לקבוע באיזה שלב תתקיים הפסיקה להלן האפשרויות:

Pin.IRQ_FALLING - interrupt on falling edge.

Pin.IRQ_RISING - interrupt on rising edge.

Pin.IRQ_LOW_LEVEL - interrupt on low level.

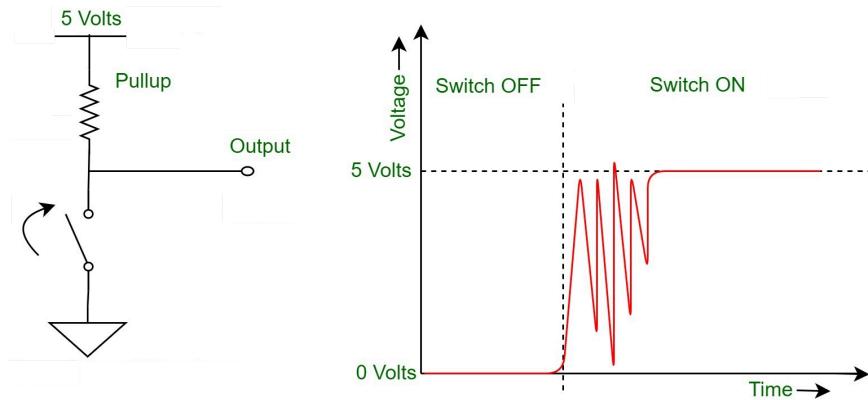
Pin.IRQ_HIGH_LEVEL - interrupt on high level.

זהירות!! שימוש לב שבגלא בעית ריטוטים בכל לחיצה על הלחצן נקלט מספר הודעה על המסר. זאת למרות של לחצנו רק פעם אחת.

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
pressed  
pressed  
pressed  
pressed  
pressed
```

נדגים תרשימים חשמלי ותרשימים של אות המבוא של הדק הפסיקה:



זמן הריתוט (switch debounce) (נמשך 50ms עד 200ms)

להלן דוגמת קוד לתוכנה הממתינה לחיצה על לחצן המחבר להדק 33 של הבקר. ברגע לחיצה יופעל טיימר למשך 200ms לתקן טיפול בעית הריטוטים ולאחר מכן תופעל הפעולה on_pressed שתציג על המסך את ההודעה "pressed"

```
from machine import Pin, Timer

def on_pressed(pin):
    print('pressed')

def debounce(pin):
    # Start or replace a timer for 200ms, and trigger on_pressed.
    timer.init(mode=Timer.ONE_SHOT, period=200, callback=on_pressed)

# Register a new hardware timer.
timer = Timer(0)
```

```

# Setup the button input pin with a pull-up resistor.

button = Pin(33, Pin.IN, Pin.PULL_UP)

# Register an interrupt on rising button input.

button.irq(debounce, Pin.IRQ_RISING)

```

ניתן לראות מנגנון פסיקה המפעיל טיימר למשך 200ms שבסופה מופעלת הפעולה .on_pressed

נדגים קוד, לא ממש יעיל, לטיפול במספר לחיצים:

```

from machine import Pin, Timer

def on_pressed1(timer):
    print('PIN 26 pressed')

def on_pressed2(timer):
    print('PIN 25 pressed')

def on_pressed3(timer):
    print('PIN 33 pressed')

def debounce1(pin):
    timer.init(mode=Timer.ONE_SHOT, period=200, callback=on_pressed1)

def debounce2(pin):
    timer.init(mode=Timer.ONE_SHOT, period=200, callback=on_pressed2)

def debounce3(pin):
    timer.init(mode=Timer.ONE_SHOT, period=200, callback=on_pressed3)

timer = Timer(0)

button1 = Pin(26, Pin.IN, Pin.PULL_UP)
button2 = Pin(25, Pin.IN, Pin.PULL_UP)
button3 = Pin(33, Pin.IN, Pin.PULL_UP)

```

```

button1.irq(debounce1, Pin.IRQ_RISING)
button2.irq(debounce2, Pin.IRQ_RISING)
button3.irq(debounce3, Pin.IRQ_RISING)

```

לצורך שימוש קוד יעיל לטיפול בלחצים המחברים לפוסיקות חומרה נממש מחלקה בשם Button לטיפול בלחץ כללי תוך שימוש בפוסיקה:

```

import time
from micropython import const
from machine import Pin, Timer

BUTTON_A_PIN = const(26)
BUTTON_B_PIN = const(25)
BUTTON_C_PIN = const(33)

led1=Pin(13,Pin.OUT)
led2=Pin(12,Pin.OUT)
led3=Pin(14,Pin.OUT)
led4=Pin(27,Pin.OUT)

class Button:
    """
    Debounced pin handler
    usage e.g.:
    def button_callback(pin):
        print("Button (%s) changed to: %r" % (pin, pin.value()))
    button_handler = Button(pin=Pin(32, mode=Pin.IN, pull=Pin.PULL_UP),
                           callback=button_callback)
    """

    def __init__(self, pin, callback, trigger=Pin.IRQ_FALLING, min_ago=300):
        self.callback = callback
        self.min_ago = min_ago

        self._blocked = False
        self._next_call = time.ticks_ms() + self.min_ago

        pin.irq(trigger=trigger, handler=self.debounce_handler)

    def call_callback(self, pin):
        self.callback(pin)

```

```

def debounce_handler(self, pin):
    if time.ticks_ms() > self._next_call:
        self._next_call = time.ticks_ms() + self.min_ago
        self.call_callback(pin)
    else:
        #    print("debounce: %s" % (self._next_call - time.ticks_ms()))

def button_a_callback(pin):
    print("Button A (%s) changed to: %r" % (pin, pin.value()))
    led1.value(1)
    led2.value(1)
    led3.value(1)
    led4.value(1)

def button_b_callback(pin):
    print("Button B (%s) changed to: %r" % (pin, pin.value()))
    led1.value(0)
    led2.value(0)
    led3.value(0)
    led4.value(0)

def button_c_callback(pin):
    print("Button C (%s) changed to: %r" % (pin, pin.value()))

button_a = Button(pin=Pin(BUTTON_A_PIN, mode=Pin.IN, pull=Pin.PULL_UP),
                  callback=button_a_callback)

button_b = Button(pin=Pin(BUTTON_B_PIN, mode=Pin.IN, pull=Pin.PULL_UP),
                  callback=button_b_callback)

button_c = Button(pin=Pin(BUTTON_C_PIN, mode=Pin.IN, pull=Pin.PULL_UP),
                  callback=button_c_callback)

```

מקורות:

<https://gist.github.com/jedie/8564e62b0b8349ff9051d7c5a1312ed7>

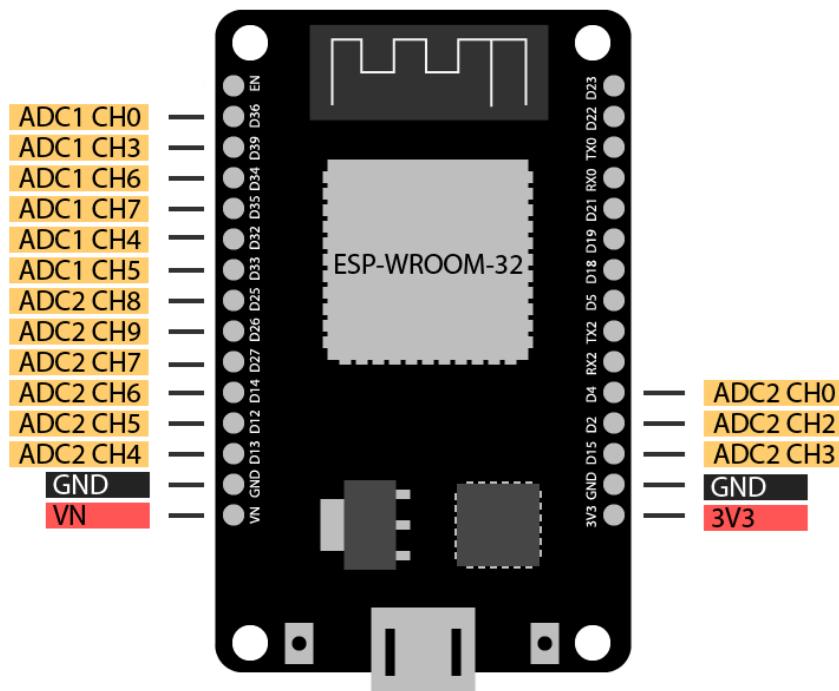
משימה 6 - קלט אות אולוגי

קישורים:

<https://randomnerdtutorials.com/esp32-esp8266-analog-readings-micropython/>

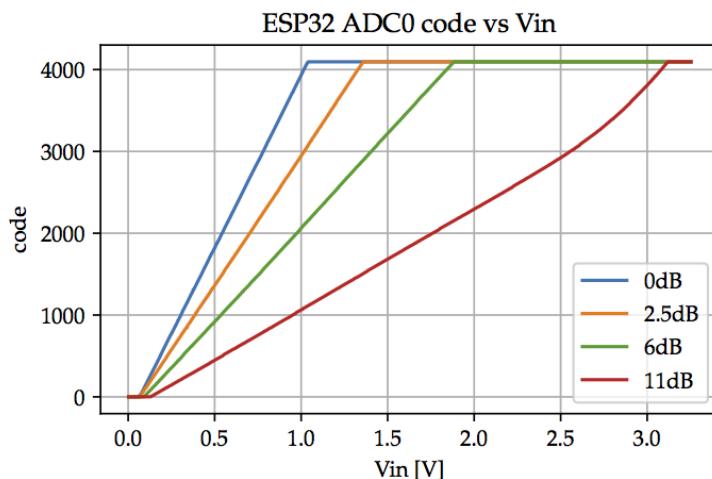
בקר ESP32 כולל שני ממירים ADC. הראשון ADC1 כולל 8 מבואות כאשר רק 6 מהם זמינים לעובדה. השני ADC2 כולל 10 מבואות. להלן רשימת המבואות:

- ADC1_CH0 :GPIO 36
- ADC1_CH1 :GPIO 37 (NOT AVAILABLE)
- ADC1_CH2 :GPIO 38 (NOT AVAILABLE)
- ADC1_CH3 :GPIO 39
- ADC1_CH4 :GPIO 32
- ADC1_CH5 :GPIO 33
- ADC1_CH6 :GPIO 34
- ADC1_CH7 :GPIO 35
- ADC2_CH0 :GPIO 4
- ADC2_CH1 :GPIO 0
- ADC2_CH2 :GPIO 2
- ADC2_CH3 :GPIO 15
- ADC2_CH4 :GPIO 13
- ADC2_CH5 :GPIO 12
- ADC2_CH6 :GPIO 14
- ADC2_CH7 :GPIO 27
- ADC2_CH8 :GPIO 25
- ADC2_CH9 :GPIO 26



ברירת המחדל של הבקר היא המרת אות אנלוגי לדיגיטלי בטוווח שבין 0 ל- 3.3V המומר למספר בין 0 ל- 4095 (כלומר מדובר בממיר 12bit).

להדק ADC של בקר ESP32 אין התנהגות לינארית. סביר להניח שלא תוכל להבחן בין 0 ל-0.1V, או בין 3.2V ל-3.3V. לכן צריך לזכור את זה בעת השימוש ב-ADC. ניתן לראות זאת באIOR הבא:



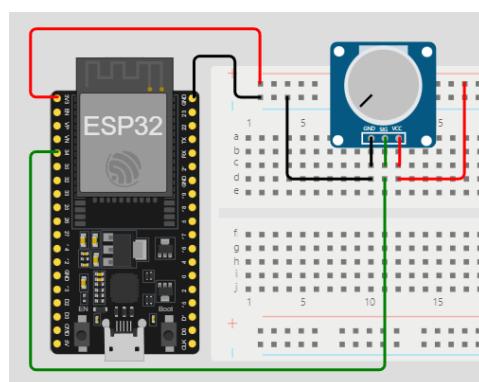
כמו כן ניתן לראות שאפשר לקבוע את טווח ההמרה של הממיר על פ' הגדרה שניתן לעשوت בחומרה:

ADC.ATTN_0DB: 0dB attenuation, gives a maximum input voltage of 1.00v

ADC.ATTN_2_5DB: 2.5dB attenuation, gives a maximum input voltage of approximately 1.34v

ADC.ATTN_6DB: 6dB attenuation, gives a maximum input voltage of approximately 2.00v

ADC.ATTN_11DB: 11dB attenuation, gives a maximum input voltage of approximately 3.6v



להלן דוגמה לעבודה עם המmir:

```
from machine import Pin, ADC
adc_pin = Pin(34, mode=Pin.IN)
adc = ADC(adc_pin)
adc.atten(ADC.ATTN_11DB)
print(adc.read())
```

ההוראה השנייה בקוד יוצרת עצם מהמחלקה PIN בשם pin_adc. עצם זה מועבר כפרמטר לפעולה הבונה של המחלקה ADC כדי ליצור את העצם adc שאיתו נעבד כדי לקרוא את הערך האנלוגי.

כמובן ניתן לכתוב אותו הקוד בכתביה מקוצרת כך:

```
from machine import Pin, ADC

adc = ADC(Pin(34)) # Create an ADC object linked to pin 34
adc.atten(ADC.ATTN_11DB)
print(adc.read())
```

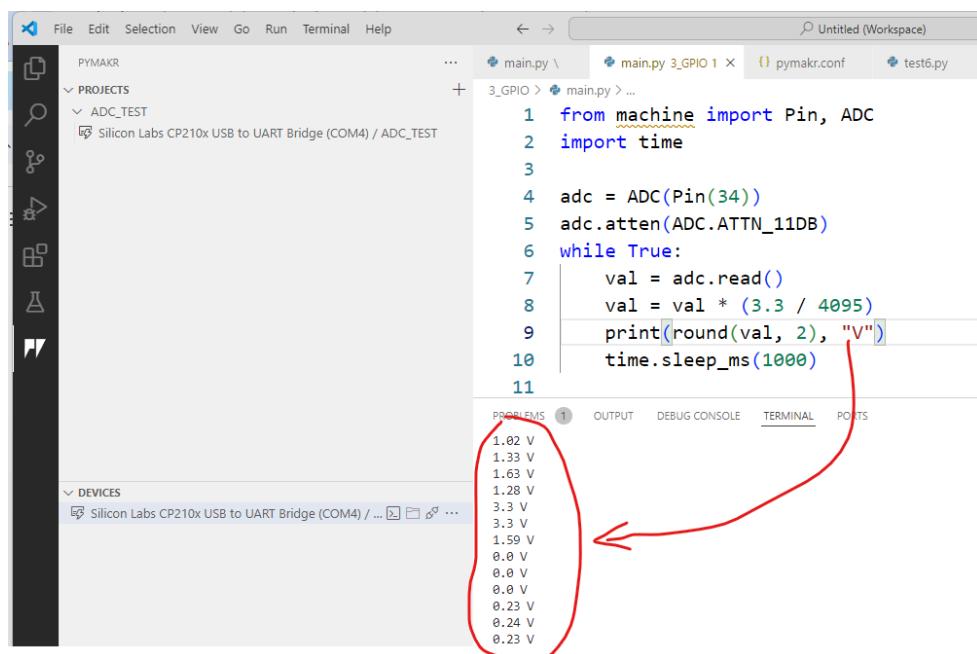
דוגמת הקוד הבא מדגימה כיצד להמיר את הערך המתתקבל למתח:

```
from machine import Pin, ADC
import time

adc = ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)

while True:
    val = adc.read()
    val = val * (3.3 / 4095)
    print(round(val, 2), "V")
    time.sleep_ms(1000)
```

נקבל את הפלט הבא:



The screenshot shows the PyMakr IDE interface. On the left is a sidebar with icons for projects, devices, and other tools. The main workspace shows a file named 'main.py' with the following code:

```
from machine import Pin, ADC
import time

adc = ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)

while True:
    val = adc.read()
    val = val * (3.3 / 4095)
    print(round(val, 2), "V")
    time.sleep_ms(1000)
```

Below the code editor is a terminal window titled 'Untitled (Workspace)' showing a list of voltage values. A red circle highlights the first few values, and a red arrow points from the terminal window back up to the code editor, indicating that the code is running and outputting the results shown in the terminal.

Voltage (V)
1.02 V
1.33 V
1.63 V
1.28 V
3.3 V
3.3 V
1.59 V
0.0 V
0.0 V
0.0 V
0.23 V
0.24 V
0.23 V

מהדוגמה ניתן לראות שהרזולוציה של ממיר ה-ADC היא 12bit . בקר ESP32 מאפשר את שינוי רזולוציית המmir.

כברירת מחדל, התוכניתת תפעל ADC של 12 סיביות ונותנת לנו ערכים בין 0-4095 אך אנו יכולים לשנות את הרזולוציה ב-ESP32 בהתאם לצרכים שלנו. ניתן להציג זאת על ידי שימוש בפונקיה width באופן הבא:

```
adc.width(ADC.WIDTH_9BIT)      #For range between 0-511
adc.width(ADC.WIDTH_10BIT)     #For range between 0-1023
adc.width(ADC.WIDTH_11BIT)     #For range between 0-2047
adc.width(ADC.WIDTH_12BIT)     #For range between 0-4095
```

נבחן זאת דרך הדוגמה הבא:

```
from machine import Pin, ADC
import time

adc = ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_9BIT) #For range between 0-511
while True:
    val = adc.read()
    val = val * (3.3 / 511)
    print(round(val, 2), "V")
    time.sleep_ms(1000)
```

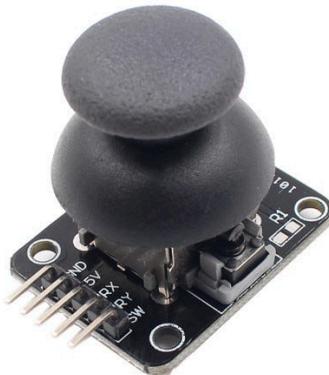
נקבל את הפלט הבא:

```
1  from machine import Pin, ADC
2  import time
3
4  adc = ADC(Pin(34))
5  adc.atten(ADC.ATTN_11DB)
6  adc.width(ADC.WIDTH_9BIT) #For range between 0-511
7
8  while True:
9      val = adc.read()
10     val = val * (3.3 / 511)
11     print(round(val, 2), "V")
12     time.sleep_ms(1000)
13
```

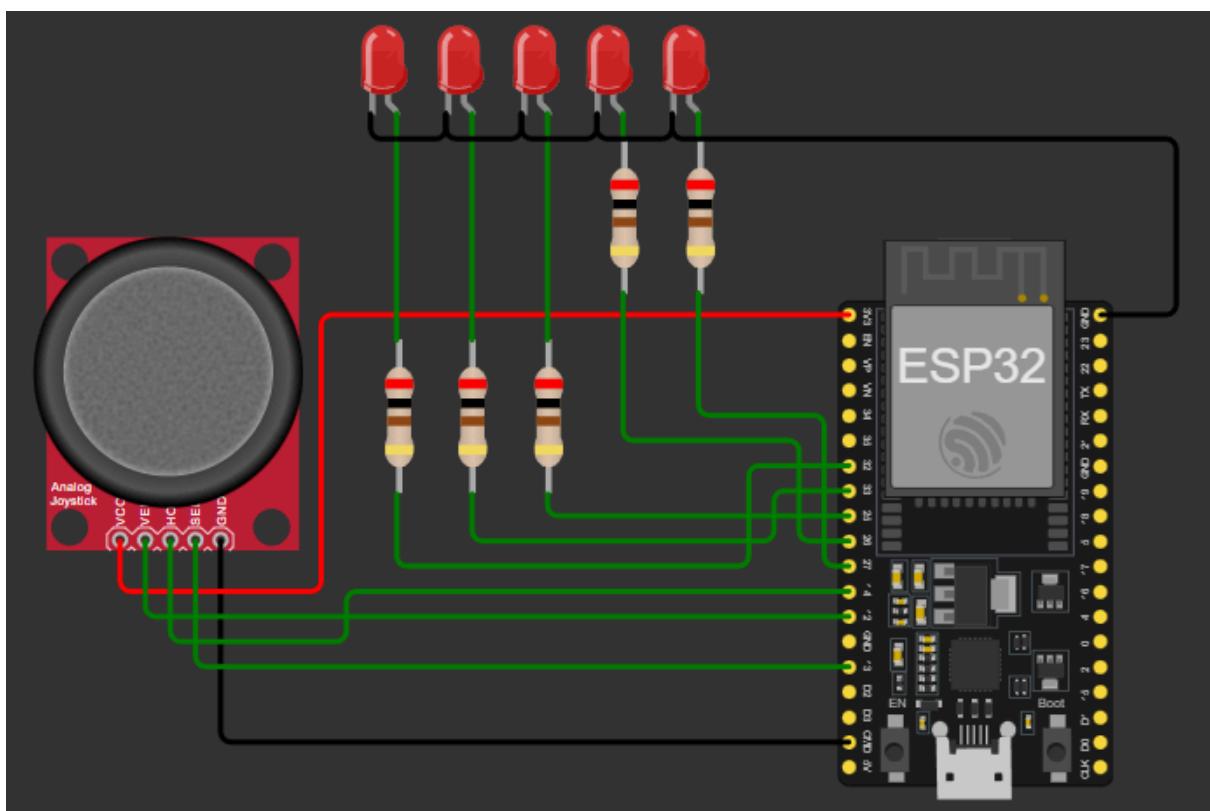
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

2.52 V
2.03 V
1.5 V
1.01 V
0.14 V
0.14 V
0.0 V
0.0 V

יישום analog-joystick במבוא אנלוגי



להלן חיבורו המugenל



להלן דוגמת קוד שמציצה את הנוריות ימינה ושמאליה בהתאם לתזוזת ה- joystick

```
from machine import Pin, ADC
from time import sleep

#joystick setup-----
sw = Pin(13,Pin.IN, pull= Pin.PULL_UP)
x_pin = Pin(12,Pin.IN)
y_pin = Pin(14,Pin.IN)

x = ADC(x_pin)
y = ADC(y_pin)
```

```

x.attenuation(ADC.ATTN_11DB)
y.attenuation(ADC.ATTN_11DB)
#joystick setup-----


#leds setup-----
led1 = Pin(27,Pin.OUT)
led2 = Pin(26,Pin.OUT)
led3 = Pin(25,Pin.OUT)
led4 = Pin(33,Pin.OUT)
led5 = Pin(32,Pin.OUT)

led_arr = [led1,led2,led3,led4,led5]
active_led = [1,0,0,0,0]
#leds setup-----


while True :

    if x.read() <= 500:
        active_led = active_led[1:] + active_led[:1]
    elif x.read() >= 2200:
        active_led = active_led[4:] + active_led[:4]

    for i in range(len(led_arr)):
        led_arr[i].value(active_led[i])

    #print("x= " ,x.read(), "    y=" ,y.read() , "    sw=" , sw.value())
    sleep(0.1)

```

<https://wokwi.com/projects/411904837659095041>

דוגמה לפעולות הZZה שמאליה של איבר ברשימה:

```

active_led = [1,0,0,0,0]
print(active_led[1:])
print(active_led[:1])
active_led = active_led[1:] + active_led[:1]
print(active_led)

[0, 0, 0, 0]
[1]
[0, 0, 0, 0, 1]

```

דוגמה לפעולות הZZה ימינה של איבר ברשימה:

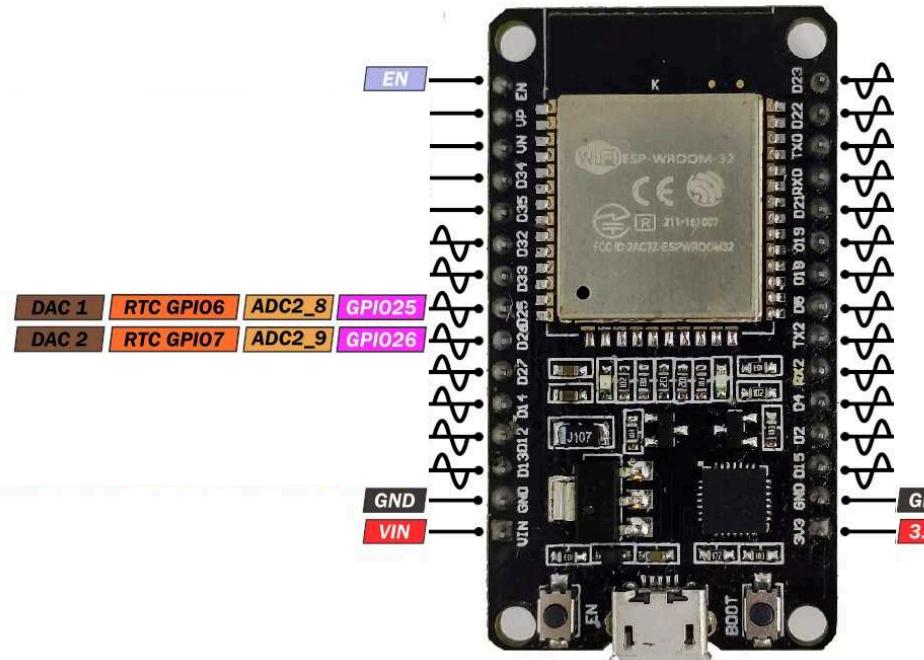
```
active_led = [1,0,0,0,0]
print(active_led[4:])
print(active_led[:4])
active_led = active_led[4:] + active_led[:4]
print(active_led)
```

```
[0]  
[1, 0, 0, 0]  
[0, 1, 0, 0, 0]
```

תודה לאריאל הרמן.

משימה 7 - ממיר דיגיטלי לאנלוגי DAC

בקר ESP32 מספק לנו 2 ממירים דיגיטליים-לאנלוגיים (DAC) בני 8 כל אחד לצורך המרת ערכיהם דיגיטליים למתחים אנלוגיים. מתחים אלה מסופקים דרך הדקן פלט GPIO של ערוץ DAC כמפורט באירוע



ה-DAC כולל 8 סיביות מבוא כך שהערך הקלט שלו בין 0 ל-255. המתח האנלוגי שווה ערך מ-0V למתח הייחוס (Vref) כאשר בReLU המודול היא 3.3V יש רשות נגדים פנימית שימושת ברמת מתח קרוביה למתח הייחוס האספקה כמתוך הייחוס המבוא (Vref). מתח הייחוס 'Vref' מתתקבל מהפין VDD3P3_RTC של שבב2 ESP32.

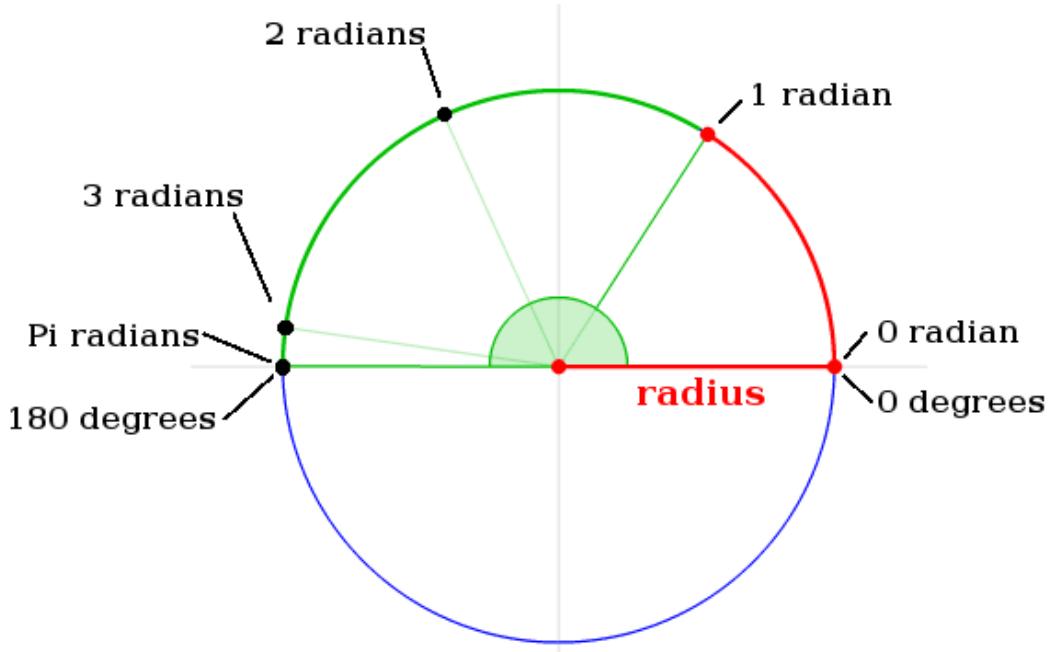
Name	No.	Type	Function
VDD3P3_RTC	19	P	Input power supply for RTC IO (2.3 V ~ 3.6 V)

דוגמם קוד בסיסי להפעלת ה-DAC:

```
from machine import DAC, Pin , reset
import time
dac1 = DAC(Pin(25))
for i in range(10):
    dac1.write(0)
    print("0V")
    time.sleep_ms(3000)
    dac1.write(128)
    print("1.65V")
    time.sleep_ms(3000)
    dac1.write(255)
    print("3.3V")
    time.sleep_ms(3000)
reset()
```

יצירת אות סינוס

לצורך יצירת אות סינוס נעשה חזרה על הבדל בין ערך הזווית במעלה לבין ערך זווית ברדיאנים



נדגים זאת בקוד:

```
import math

print("converts radians to degrees.")
radians = math.pi/2
degrees = radians * 180 / math.pi
print("radians",radians,"--> degrees",degrees)
print(math.sin(radians))

print("converts radians to degrees.")
print(math.degrees(0))
print(math.degrees(math.pi/2))
print(math.degrees(math.pi))
print(math.degrees(math.pi*1.5))

print("converts degrees to radians.")
print(math.sin(math.radians(0)))
print(math.sin(math.radians(90)))
print(math.sin(math.radians(180)))
print(math.sin(math.radians(270)))
```

נקבל את הפלט הבא:

```

>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
converts radians to degrees.
radians 1.570796 --> degrees 90.0
1.0
converts radians to degrees.
0.0
90.0
180.0
270.0
converts degrees to radians.
0.0
1.0
-8.742278e-08
-1.0

>>>

```

להלן קוד המדגים יצירת אות סינוס (גרסת 1):

```

from machine import DAC, Pin, reset
import time

dac = DAC(Pin(25))

wave = [
    0x80, 0x83, 0x87, 0x8A, 0x8E, 0x91, 0x95, 0x98, 0x9B, 0x9E, 0xA2, 0xA5,
0xA7, 0xAA, 0xAD, 0xAF,
    0xB2, 0xB4, 0xB6, 0xB8, 0xB9, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF, 0xBF, 0xBF,
0xC0, 0xBF, 0xBF, 0xBF,
    0xBE, 0xBD, 0xBC, 0xBB, 0xB9, 0xB8, 0xB6, 0xB4, 0xB2, 0xAF, 0xAD, 0xAA,
0xA7, 0xA5, 0xA2, 0x9E,
    0x9B, 0x98, 0x95, 0x91, 0x8E, 0x8A, 0x87, 0x83, 0x80, 0x7C, 0x78, 0x75,
0x71, 0x6E, 0x6A, 0x67,
    0x64, 0x61, 0x5D, 0x5A, 0x58, 0x55, 0x52, 0x50, 0x4D, 0x4B, 0x49, 0x47,
0x46, 0x44, 0x43, 0x42,
    0x41, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x41, 0x42, 0x43, 0x44,
0x46, 0x47, 0x49, 0x4B,
    0x4D, 0x50, 0x52, 0x55, 0x58, 0x5A, 0x5D, 0x61, 0x64, 0x67, 0x6A, 0x6E,
0x71, 0x75, 0x78, 0x7C
]

delay_us = 1

try:
    while True:
        for value in wave:
            dac.write(value)
            time.sleep_us(delay_us)

except KeyboardInterrupt:
    print('Ctrl-C pressed... exiting')
    reset()

```

להלן קוד המדגים ייצור סינוס (גרסת 2):

```
from machine import DAC, Pin, reset
import time

dac = DAC(Pin(25))

wave = [ 0 , 0.4818 , 0.8443 , 0.998 , 0.9048 , 0.5878 , 0.1253 , -0.3681 ,
-0.7705 ,
-0.9823 , -0.9511 , -0.6845 , -0.2487 , 0.2487 , 0.6845 , 0.9511 , 0.9823 ,
0.7705 ,
0.3681 , -0.1253 , -0.5878 , -0.9048 , -0.998 , -0.8443 ,-0.4818]

delay_us = 1

try:
    while True:
        for value in wave:
            dac.write(int((value + 1) * 127.5))
            time.sleep_us(delay_us)

except KeyboardInterrupt:
    print('Ctrl-C pressed...exiting')
    reset()
```

הסביר הקוד:

הערכים ברשימה הם בין -1 לBIN 1 לאורך מחזור סינוס מלא (0 עד $\pi/2$). לדוגמה:

$$\sin(0) = 0 \quad \sin(\pi/2) = 1 \quad \sin(\pi) = 0 \quad \sin(3\pi/2) = -1$$

הוספה 1 למשווה מזיצה את הטווח מ-[1,-1] ל-[0,2]. זה חשוב כי DAC לא מקבל ערכים שליליים ואנו חנכו רצים להמיר את גל הסינוסegal שמתחליב-0.

לבסוף מכפילים הכל ב-127.5 כי DAC של ESP32 תומך בערכים של 8 ביט, כלומר מטווח של 0–255.

אך כדי להמיר את טווח [0,2] לטווח [0,255], כופלים ב-127.5:

להלן קוד המדגים ייצור סינוס (גרסת 3):

```
from machine import DAC, Pin, reset
import time
import math

dac = DAC(Pin(25))

samples = 50
wave = []
for i in range(samples):
    angle = 2 * math.pi * i / samples
    wave.append(int((math.sin(angle) + 1) * 127.5))

delay_us = 1
```

```

try:
    while True:
        for value in wave:
            dac.write(value)
            time.sleep_us(delay_us)

except KeyboardInterrupt:
    print('Ctrl-C pressed... exiting')
    reset()

```

הסביר ההוראה:

```
angle = 2 * math.pi * i / samples
```

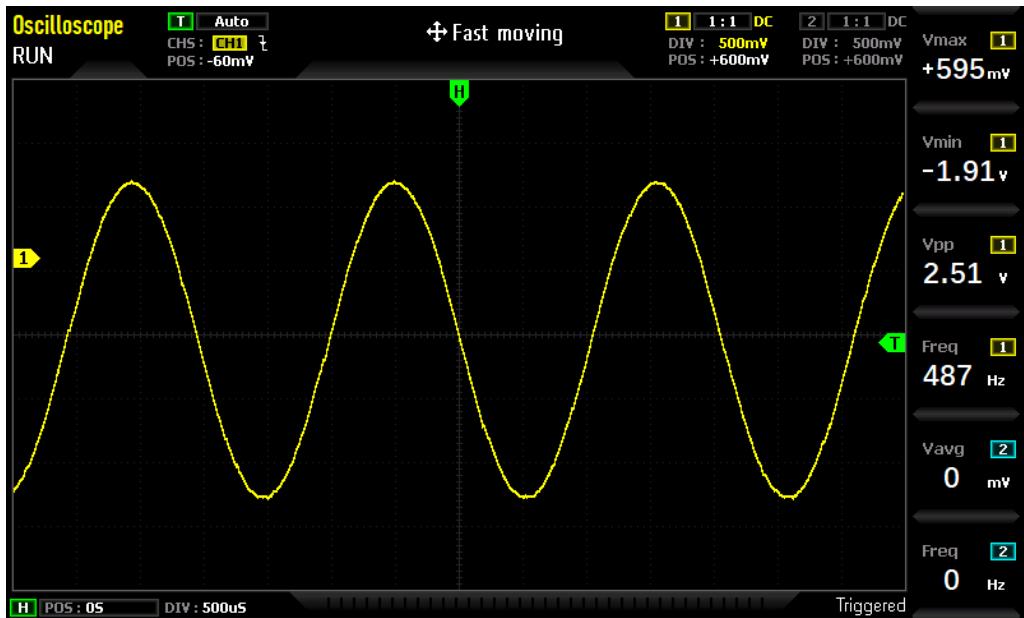
- $\text{math.pi} = \text{קבוע}$
- $2 * \text{math.pi} = \text{מעגל שלם ברדיאנים } (2\pi = 360^\circ)$
- $i = \text{אינדקס הדוגמיה הנוכחית } (0, 1, \dots, \text{samples} - 1)$
- $\text{samples} = \text{מספר הדוגמאות בכל מחזור}$

gal_sinosi הוא פונקציה מוחזקת המתוארת ע"י ($\alpha \sin \alpha$) אשר α זו **הזווית** (angle) ברדיאנים. כדי ליצור מוחזר שלם של gal_sinosi נרצה לעבור מזוית 0 עד 2π כלומר מ-0 עד 360 מעלות.

נחלק את המוחזר ל-samples חלקים שווים וכך לכל דוגמיה נמחשב את הזווית כז:

$$\frac{2\pi \cdot i}{samples} = \alpha$$

יצירת אות סינווי



יצירת שני אוטות יחד

מעשה שימוש בשני ממיר ה- DAC ליצירת שני אותו באותו תדר ובאותו וואצמה. האחד בתזוזת מופע של 90 מעלות אחד מהשני. נדגים זאת על ידי הקוד הבא:

```
from machine import Pin, DAC
import math

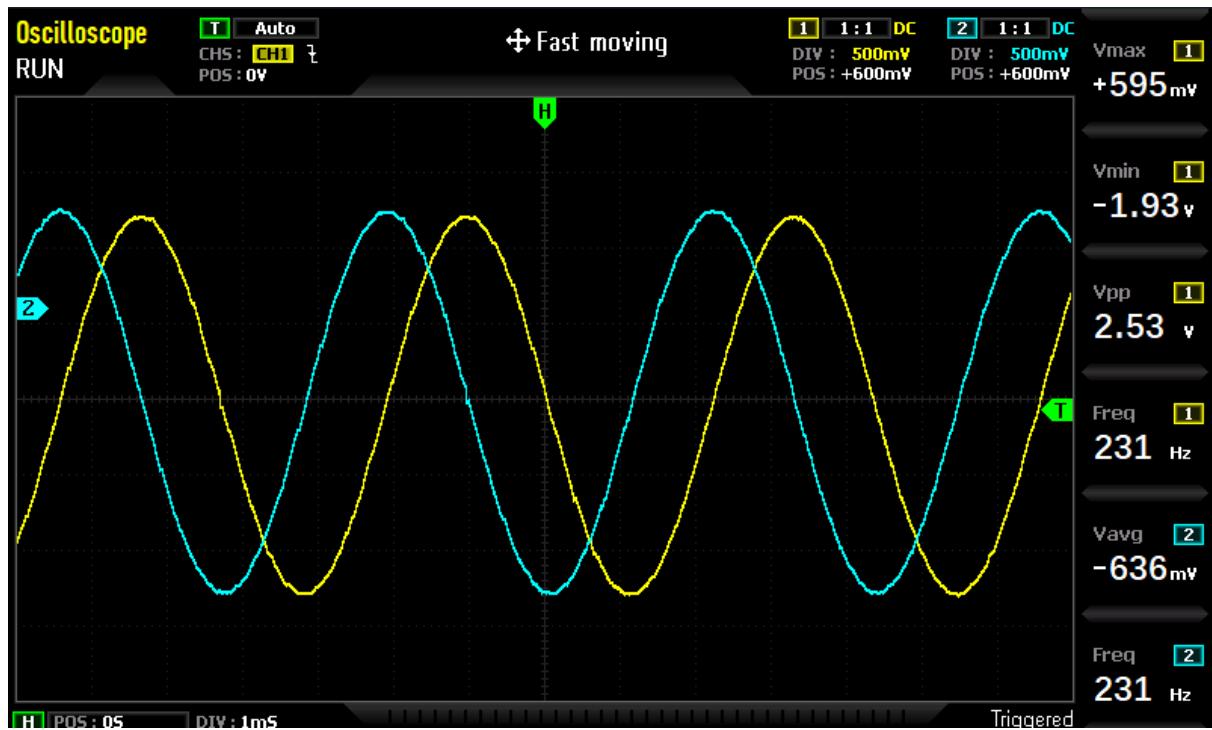
dac1 = DAC(Pin(25, Pin.OUT))
dac1.write(0)
dac2 = DAC(Pin(26, Pin.OUT))
dac2.write(0)

# global variables
P = 2
Q = 2
N = 200
A = 100
Omega = 2*math.pi/N

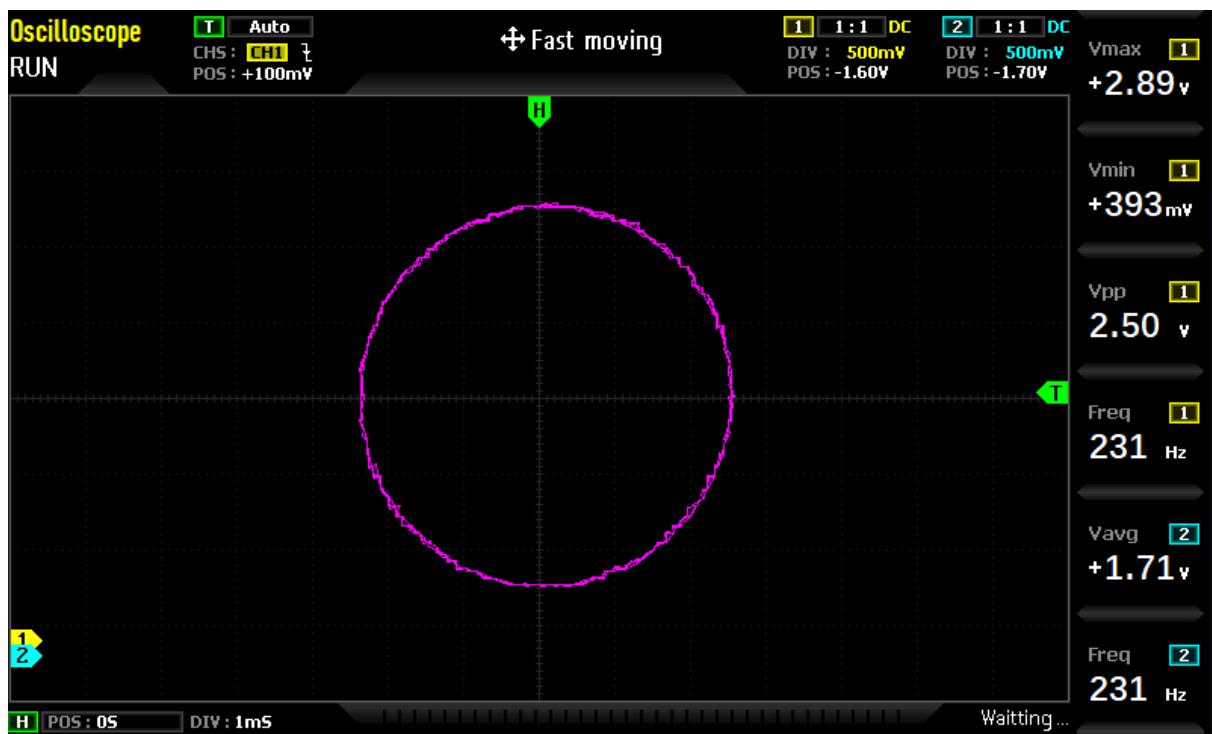
sin_table = []
cos_table = []
for i in range(N):
    arg = Omega*i
    x = A*math.sin(P*arg) + 127
    y = A*math.cos(Q*arg) + 127
    sin_table.append(int(x))
    cos_table.append(int(y))

index = 0 # index range: 0..(N-1)
while True:
    x = sin_table[index]
    y = cos_table[index]
    index = (index+1) % N
    dac1.write(x)
    dac2.write(y)
```

נציג את שני האוטות על גבי סקופ (ערוץ אחד מחובר להדק 25 והערוץ השני של הסקופ מחובר להדק 26) נקבל את האות הבא:



כמובן שאפשר לעבור למצב XY כדי לקבל את הצורה הנ"ל:



תרגיל:

דוגמת קוד המיצרת צלילים סינוסואידליים עבור אוקטבה אחת

```
from machine import DAC, Pin, reset
import time
import math

dac = DAC(Pin(25))

notes = [
    523,    # C (Octave 5)
    587,    # D
    659,    # E
    698,    # F
    784,    # G
    880,    # A
    988,    # B
    1046   # C (Octave 6)
]

samples = 25
wave = []
for i in range(samples):
    angle = 2 * math.pi * i / samples
    wave.append(int((math.sin(angle) + 1) * 127.5)) # 0-255

def play_tone(freq, duration_ms=300):
    delay_us = int(1000000 / (freq * samples))
    end_time = time.ticks_ms() + duration_ms
    while time.ticks_ms() < end_time:
        for val in wave:
            dac.write(val)
            time.sleep_us(delay_us)

    for i in range(8):
        play_tone(notes[i])
    for i in range(8):
        play_tone(notes[i], duration_ms=1000)

reset()
```

יש לחבר רמקול להדק 25 ולשמעו את הצלילים.

הסביר ההוראה:

```
delay_us = int(1000000 / (freq * samples))
```

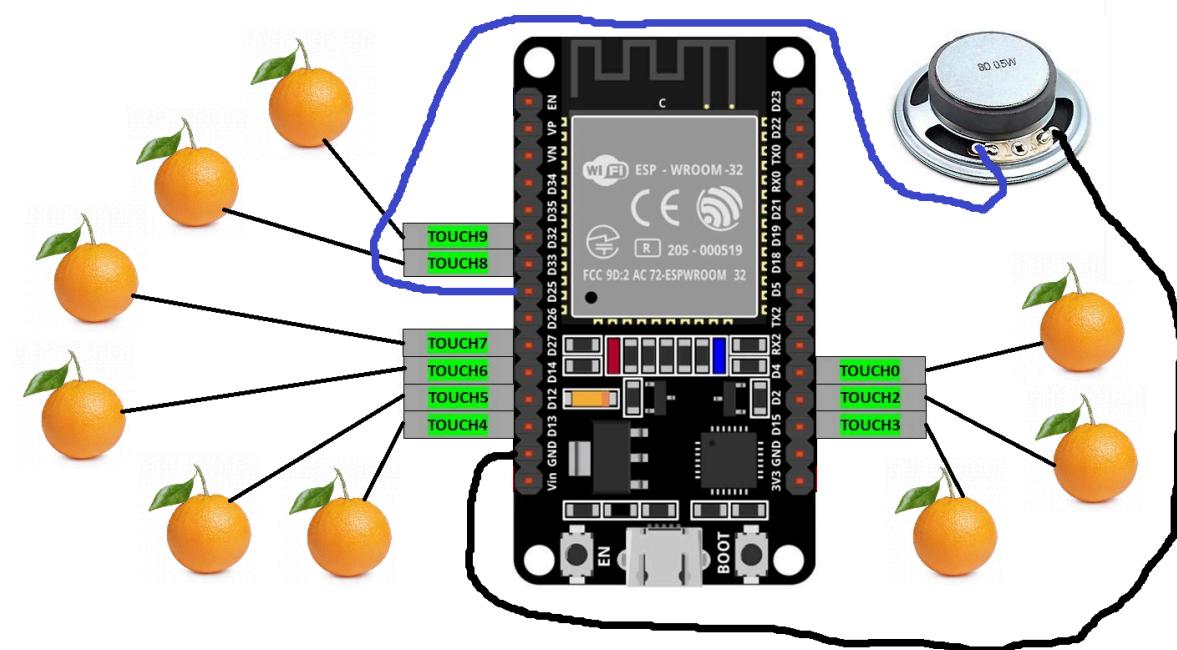
- $\text{delay_us} = \text{int}(1000000 / (\text{freq} * \text{samples}))$ = התדר של הצליל (ביחיות הרץ, כלומר מחזורים לשנייה)
- samples = מספר הדגימות בכל מחזור של האgel
- $1000000 = \text{מספר המיקרו-שניות בשניה} (1 \text{ שניה} = 1,000,000 \text{ מיקרו-שניות})$
- delay_us = ההשניה (delay) בין כל דוגימה ודוגימה, במיקרו-שניות

לדוגמא: אם רצים לשמוע גל סינוס בתדר מסוים, למשל $523\text{Hz} = \text{freq}$, צריך להוציא את כל הדגימות של מחזור סינוס אחד **523 פעמיים בשניה**.

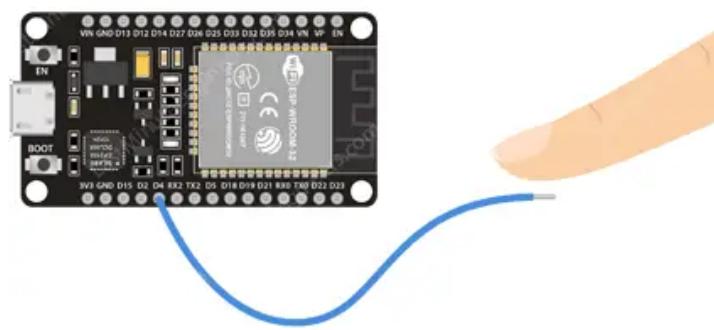
לכן אם יש לנו **100 דגימות ב-samples**, זה אומר בשניה אחת נצרך להוציא $100 \times 523 = 52,300$ דגימות. כלומר כל דגימה צריכה לצאת אחת ל- 19 מיקרו שניות.

$$\frac{1,000,000}{100 \cdot 523} = 19\mu\text{SEC}$$

יש לחבר 8 דק' Touch כדי ליצור ארגן פירוט כמתואר בתמונה הבאה:



תזכורת: חיישנים קיבוליים ב-ESP32 פועלים על ידי מדידת שינוי הקיבול החשמלי של הדק. כאשר האצבע שלנו **מתקרבת** או מגעת בדק, מתרחש שינוי בקיבול עקב התכונות החשמליות של גוף האדם. ה-ESP32 מודד את הזמן שלוקח להדק להיטען או להתרפרק, מדידה זו משתנה בהתאם לקיבול שנוצר בין הבדיקה לבין הגוף שלנו. הערך המוחזר הוא מספר המיצג את זמן הפעינה/פריקה - ערך נמוך יותר מאשר מגע.



ל-ESP32 יש מספר דק' מגע קיבוליים (T0, T1, T2, T3, T4, T5, T6, T7, T8, T9). לא כל הדקים זמינים בכל לוחות ESP32, יש לבדוק את מסמכי הלוח הספציפי.

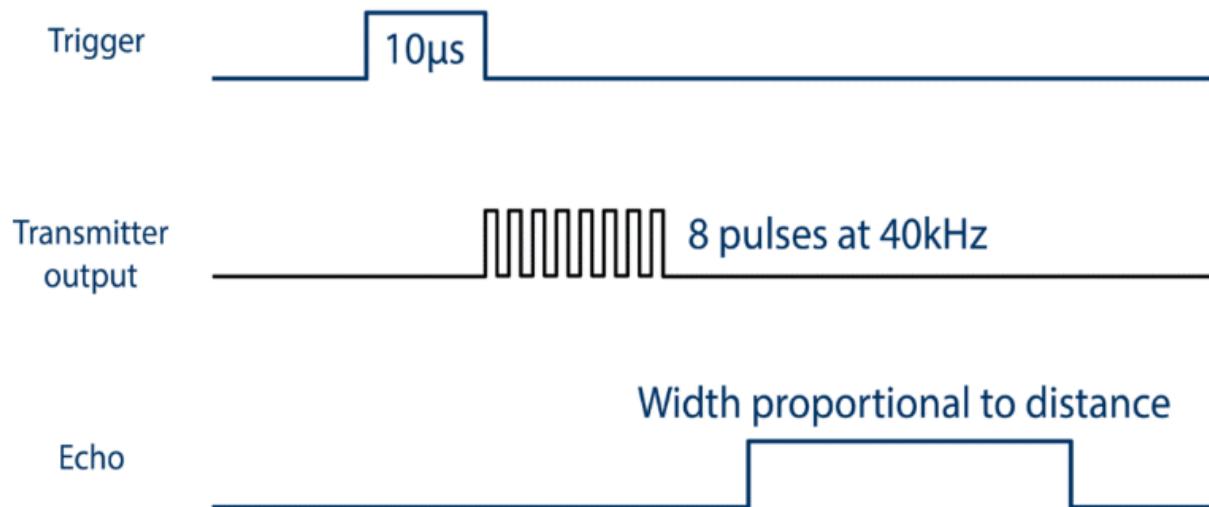
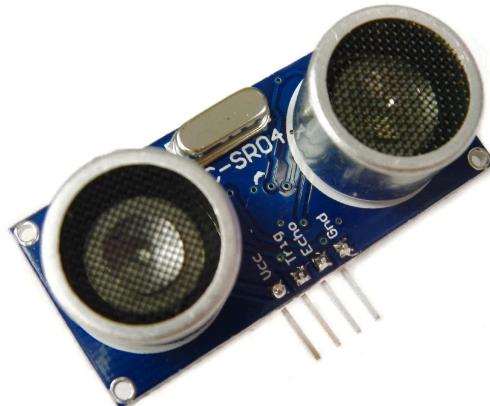
משימה 8 - הפעלת מד מרחק אולטראוני דגם hc-sr04

קישורים:

<https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi>

ה-HC-SR04 הוא מד מרחק אולטרו סוני זול ונפוץ מאוד. לרכיב ארבע הדקים:

- הדק Vcc המתקבל 5V.
- הדק אדמה - Gnd.
- הדק Trigger - המשמש להפעלת המדידה.
- הדק Echo - שדרוכה אנו מקבלים מדידת מרחק כפולו שהרכיב שלו יחסית למרחק.



הבעיה המרכזי בחיבור מד מרחק hc-sr04 לבודר ESP-32 היא ההבדל בין מתח העבודה של 2 הרכיבים. בעוד hc-sr04 עובד במתח עבודה של 5V, מתח העבודה של ESP-32 הוא 3.3V ממשמע שאנו זוקקים למתח אספקה חיצוני של 5V לעבוד החישון כמו כן חובה עלינו להתאים בין רמת המתחים של הדק Echo שמתקבל בرمות של 5V למתח של 3.3V אחרית אנו נגרום נזק לבקר. במצב ההופך שבו דרך Trigger המסופק בرمות של 3.3V נכנס לחישון שעבוד בرمות של 5V לא יגרום נזק.


```
# Init echo pin (in)
echo = Pin(echo_pin, mode=Pin.IN, pull=None)

# Send a 10us pulse.
trigger.value(0)
sleep_us(5)
trigger.value(1)
sleep_us(10)
trigger.value(0)

pulse_time = time_pulse_us(echo, 1, 500*2*30)
cms = (pulse_time / 2) / 29.1    # 1cm each 29.1us
return cms

for i in range(5):
    distance = Get_distance_cm()
    print('Distance:', distance, 'cm')
    time.sleep(2)
```

נקבל את הפלט הבא:

```
load:0x3fff0030,len:4728
load:0x40078000,len:14888
load:0x40080400,len:3368
entry 0x400805cc
Distance: 10.7732 cm
Distance: 13.33333 cm
Distance: 13.81443 cm
Distance: 16.09966 cm
Distance: 17.1134 cm
}
} 60
MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with ESP32
Type "help()" for more information.
>>>
```

נער בדוגמת הקוד הבא:

<https://github.com/rsc1975/micropython-hcsr04>

כדי להגדיר מחלקה המטפלת במד המרחק. את הגוד הבא נחלק ל-2 חלקיים. הראשון מחלקת ברכיב:

```
from machine import Pin, time_pulse_us
from utime import sleep_us

__version__ = '0.2.1'
__author__ = 'Roberto Sánchez'
__license__ = "Apache License 2.0.
https://www.apache.org/licenses/LICENSE-2.0"

class HCSR04:
    """
        Driver to use the ultrasonic sensor HC-SR04.
        The sensor range is between 2cm and 4m.

        The timeouts received listening to echo pin are converted to
        OSError('Out of range')
```

```

"""
# echo_timeout_us is based in chip range limit (400cm)
def __init__(self, trigger_pin, echo_pin, echo_timeout_us=500*2*30):
    """
        trigger_pin: Output pin to send pulses
        echo_pin: Readonly pin to measure the distance. The pin should be
protected with 1k resistor
        echo_timeout_us: Timeout in microseconds to listen to echo pin.
        By default is based in sensor limit range (4m)
    """
    self.echo_timeout_us = echo_timeout_us
    # Init trigger pin (out)
    self.trigger = Pin(trigger_pin, mode=Pin.OUT, pull=None)
    self.trigger.value(0)

    # Init echo pin (in)
    self.echo = Pin(echo_pin, mode=Pin.IN, pull=None)

def _send_pulse_and_wait(self):
    """
        Send the pulse to trigger and listen on echo pin.
        We use the method `machine.time_pulse_us()` to get the microseconds
until the echo is received.
    """
    self.trigger.value(0) # Stabilize the sensor
    sleep_us(5)
    self.trigger.value(1)
    # Send a 10us pulse.
    sleep_us(10)
    self.trigger.value(0)
    try:
        pulse_time = time_pulse_us(self.echo, 1, self.echo_timeout_us)
        if pulse_time < 0:
            MAX_RANGE_IN_CM = const(500)
            pulse_time = int(MAX_RANGE_IN_CM * 29.1) # 1cm each 29.1us
        return pulse_time
    except OSError as ex:
        if ex.args[0] == 110: # 110 = ETIMEDOUT
            raise OSError('Out of range')
        raise ex

def distance_mm(self):
    """
        Get the distance in milimeters without floating point operations.
    """
    pulse_time = self._send_pulse_and_wait()

    # To calculate the distance we get the pulse_time and divide it by
2
    # (the pulse walk the distance twice) and by 29.1 because
    # the sound speed on air (343.2 m/s), that It's equivalent to
    # 0.34320 mm/us that is 1mm each 2.91us

```

```

    # pulse_time // 2 // 2.91 -> pulse_time // 5.82 -> pulse_time * 100
// 582
    mm = pulse_time * 100 // 582
    return mm

def distance_cm(self):
    """
    Get the distance in centimeters with floating point operations.
    It returns a float
    """
    pulse_time = self._send_pulse_and_wait()

    # To calculate the distance we get the pulse_time and divide it by
2
    # (the pulse walk the distance twice) and by 29.1 because
    # the sound speed on air (343.2 m/s), that It's equivalent to
    # 0.034320 cm/us that is 1cm each 29.1us
    cms = (pulse_time / 2) / 29.1
    return cms

```

קוד השני מזמן את המחלקה ומציג את המרחק:

```

import time
from hcsr04 import HCSR04

sensor = HCSR04(trigger_pin=32, echo_pin=34)
for i in range(10):
    distance = sensor.distance_cm()
    print('Distance:', distance, 'cm')
    time.sleep(1.5)

```

פלט התוכנית צפוי להיראות כה:

The screenshot shows a MicroPython code editor interface. The Explorer sidebar on the left lists files: pymakr.conf, boot.py, hcsr04.py, and main.py. The main area displays the content of main.py:

```
1 import time
2 from hcsr04 import HCSR04
3
4 sensor = HCSR04(trigger_pin=32, echo_pin=34)
5 for i in range(10):
6     distance = sensor.distance_cm()
7     print('Distance:', distance, 'cm')
8     time.sleep(1.5)
```

The code uses the HCSR04 module to measure distance 10 times, printing each result in centimeters. Red annotations include:

- A red bracket on the left side of the Explorer sidebar, spanning from the top to the bottom of the list.
- A red arrow pointing from the top of the main.py code area down to the first line of the code.
- A large red brace on the right side of the output console, grouping the printed distance values.
- The number "60" written in red next to the brace.

The output console at the bottom shows the execution results:

```
rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14888
load:0x40080400,len:3368
entry 0x400805cc
Distance: 31.9244 cm
Distance: 120.2577 cm
Distance: 26.03093 cm
Distance: 14.12371 cm
Distance: 11.90722 cm
Distance: 108.2646 cm
Distance: 12.13058 cm
Distance: 9.621993 cm
Distance: 9.725086 cm
Distance: 9.75945 cm
```

משימה 9 - תקשורת UART בין 2 בקרים

קישורים:

<https://docs.micropython.org/en/latest/library/machine.UART.html>

לבקר ESP32 יש 3 ממתקי תקשורת UART (שרק אחד זמין לשימוש) על פי המפרט הבא:

UART0: (GPIO 1 and GPIO3) משמש לתקשורת מול המחשב -

UART1: (GPIO 9 and GPIO10) – connected to the ESP32 SPI flash memory, so you can't use them.

UART2: (GPIO 17 and GPIO 16) – זמין לשימוש -

למרות שהדק בירית המחדר של UART1 תפקידים ניתן לשנות את הדק בירית המחדר של הדקים אלו. ולהעיבר אותם לכל הדק GPIO , למעט הדקים 34-39 שהם הדק קלט בלבד שיכולים לשמש כ-אץ אך לא כ-אקס.

לসיכון כדי למנוע התנגדויות הדקים ל- UART1 פשוט ספקו את מספרי ההדקים של אט-אץ בעת יצירת עצם חדש של המחלקה UART בעופן הבא:

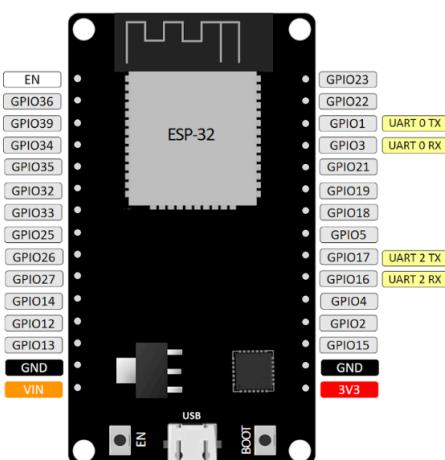
```
uart = UART(1, baudrate=9600, tx=19, rx=18)
```

ניתן לראות שלמרות שהדק בירית המחדר של UART1 הם tx=10 ו-rx=9

החלפנו אותם להדקים tx=19, rx=18.

	UART0	UART1	UART2
tx	1	10	17
rx	3	9	16

להלן מיקום הדק בירית המחדר של ה- UART בעוקה:

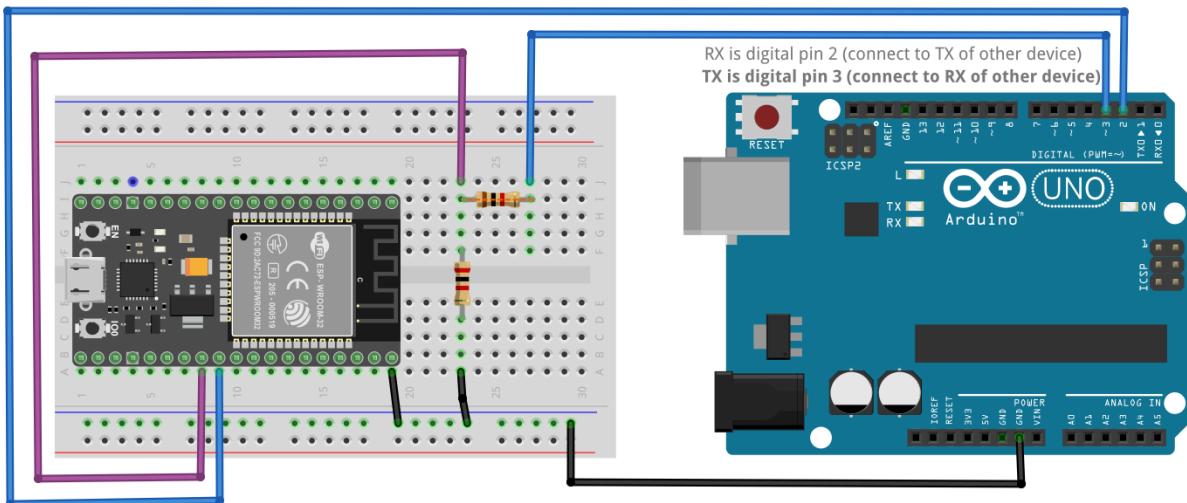
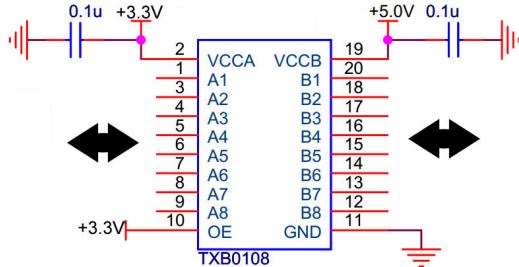


כולם UART0 לא זמין - UART2 זמין דרך הדקים tx=17 ו-rx=16.

תקשורת בין ESP32 לArduino UNO

להלן תיאור עקרוני של החיבור בין בקר UNO ESP32 לבי בקר TXS0108E

שימוש לב לבר שבקר Arduino עובד בטוווח מתחים של בין 0 ל- 5V ובקר ESP32 עובד בטוווח שבין 0 ל- 3.3V. לכן יש צורך להמיר את המוצא AD של בקר Arduino מ- 5V ל- 3.3V על ידי מחלק מתח או על ידי רכיב ייעודיTXS0108E



בקר ESP32 מכיל את הקוד הבא:

```
import time
from machine import UART

uart = UART(2, 115200)

def sender(i):
    s = 'Hello uart' + str(i) + '\n'
    uart.write(s)
    print(s)

def receiver():
    if uart.any():
        res = uart.readline()
        if res:
            print('Received', res)

i = 0
while True:
    sender(i)
    receiver()
    i += 1
```

```
time.sleep(5)
```

בקר UNO מוביל את הקוד הבא:

```
#include <SoftwareSerial.h>

// RX is digital pin 2 (connect to TX of other device)
// TX is digital pin 3 (connect to RX of other device)
SoftwareSerial ESP32Serial(2, 3); // RX, TX

void setup()
{
    // Open serial communications and wait for port to open:
    Serial.begin(115200);

    // set the data rate for the SoftwareSerial port
    ESP32Serial.begin(115200);
}

void loop() // run over and over
{
    if (ESP32Serial.available())
        Serial.write(ESP32Serial.read());
    if (Serial.available())
        ESP32Serial.write(Serial.read());
}
```

נקבל את הפלט הבא:

The screenshot shows an IDE interface with two code editors and a terminal window.

Left Editor (Python):

```
8     while True:
9         s = 'Hello uart' + str(i) + '\n'
10        # await swriter.awrite(s)
11        uart.write(s)
12        print(s)
13        await asyncio.sleep(2)
```

Right Editor (Arduino):

```
8 {
9     // Open serial communications and w
10    Serial.begin(115200);
11
12    // set the data rate for the Softwa
13    ESP32Serial.begin(115200);
14 }
15
16 void loop() // run over and over
17 {
18     if (ESP32Serial.available())
19         Serial.write(ESP32Serial.read());
```

Terminal Output:

```
Hello uart202
Hello uart203
Hello uart204
Hello uart205
Hello uart206
Hello uart207
Hello uart208
Hello uart209
Received b'Hello from arduino\n'
Hello uart210
Hello uart211
```

להלן דוגמה לקוד עבור בקר ESP32 המימוש תקשורת אסינכרונית:

```
import uasyncio as asyncio
```

```

from machine import UART
uart = UART(2, 115200)

async def sender():
    i=0
    while True:
        s = 'Hello uart' + str(i) + '\n'
        uart.write(s)
        print(s)
        await asyncio.sleep(5)
        i=i+1

async def receiver():
    sreader = asyncio.StreamReader(uart)
    while True:
        res = await sreader.readline()
        print('Recieved', res)

loop = asyncio.get_event_loop()
loop.create_task(sender())
loop.create_task(receiver())
loop.run_forever()

```

משימה

עשו השוואה בין זמן התגובה של שידור מידע מבקר Arduinoino ESP32 לבין דוגמת הקוד הסינכרונית בין דוגמת הקוד האсинכרונית!

הבדל בין str ל-bytes בשפת Python

- str מייצג טקסט.
 - bytes מייצג את הצורה המוקודת של הטקסט, למשל לצורך שידור ב-UART או כתיבה לקובץ בינהר.
- להלן טבלת השוואה בין השניים:

טיפוס	תיאור	דוגמה
str	מחרוזת של תווים (Unicode) – טקסט קרייא לבני אדם	'Hello', 'שלום עולם'
bytes	רצף של בתים (byte sequence) – נתונים בינהרים	b'Hello', b'\xd7\x9a\xd7\x9c\xd7\x95\xd7\x9d'

המרה מ-String ל-bytes

נדגים קוד לצורך>Lencode() של טקסט(utf-8) לפ' encode()

```

s = 'שלום'
b = s.encode('utf-8')

```

```
print(b) # b'\xd7\x95\xd7\x9c\xd7\x95\xd7\x9d'
```

המרה מ-`bytes` ל-`String`:

נדגַי קוד לצורך לפונק (decode) רשימת בתים חוזרת לטקסט.

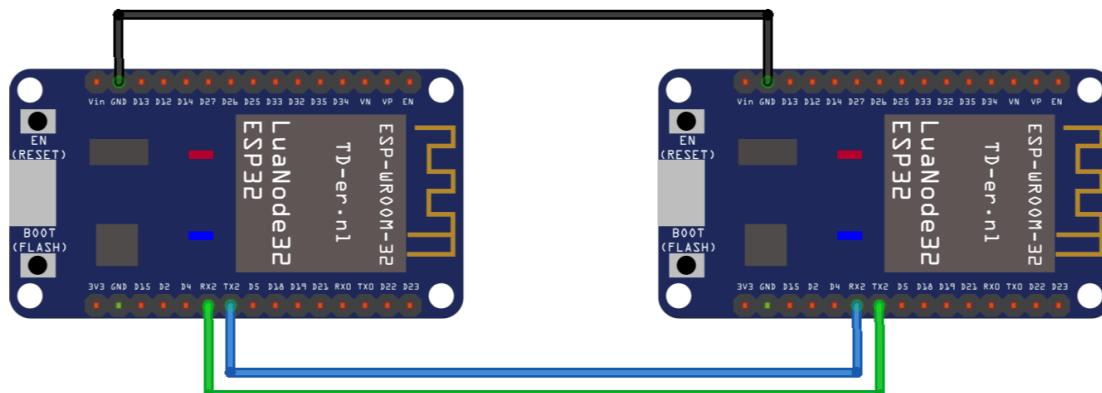
```
b = b'\xd7\x95\xd7\x9c\xd7\x95\xd7\x9d'  
s = b.decode('utf-8')  
print(s) # שלום
```

חשוב להזכיר שני הצדדים (שולח ומוקבל) צריכים להסכים על אותוקידוד לדגמה utf-8.

UTF-8 (ראשי תיבות של bit UCS Transformation Format או bit Unicode Transformation Format) הואקידוד תווים באורך משתנה ליוניקוד, שנוצר על ידי רוב פ'יק וkon למופסן. ניתן לקוד בו כל תו המצוין בתוקן יוניקוד על ידי שימוש באחד עד ארבעה בתים, תלוי בתו. הקידוד בUTF-8 מעניק את כל יתרונות השימוש בקידוד יוניקוד ומוסיף עליהם, בין היתר, גם חיסכון בזכרון, עמידות לפני איבוד או השחתת בתים ותאמיות לאחר ASCII. (ויקיפדיה)

הדגמת תקשורת בין שני בקרים ESP32

נחבר את התקשרות בין 2 הבקרים כמפורט באיר הבא:



נתען את הקוד הבא בכל אחד משני הזרים:

```
import uasyncio as asyncio  
from machine import UART  
uart = UART(2, 115200)  
  
async def sender():  
    i=0  
    while True:  
        s = 'Hello uart' + str(i) + '\n'  
        uart.write(s)  
        print(s)  
        await asyncio.sleep(2)  
        i=i+1  
  
async def receiver():  
    sreader = asyncio.StreamReader(uart)  
    while True:
```

```

        res = await sreader.readline()
        print('Recieved', res)

loop = asyncio.get_event_loop()
loop.create_task(sender())
loop.create_task(receiver())
loop.run_forever()import uasyncio as asyncio
from machine import UART
uart = UART(2, 115200)

async def sender():
    i=0
    while True:
        s = 'Hello uart' + str(i) + '\n'
        uart.write(s)
        print(s)
        await asyncio.sleep(2)
        i=i+1

async def receiver():
    sreader = asyncio.StreamReader(uart)
    while True:
        res = await sreader.readline()
        print('Recieved', res)

loop = asyncio.get_event_loop()
loop.create_task(sender())
loop.create_task(receiver())
loop.run_forever()

```

להלן דוגמה לפלט התקשרות:

```
main.py 2 x
main.py > sender
1 import uasyncio as asyncio
2 from machine import UART
3 uart = UART(2, 115200)
4
5 async def sender():
6     swriter = asyncio.StreamWriter(uart, protocol=asyncio.StreamWriterProtocol())
7     i=0
8     while True:
9         s = 'Hello uart' + str(i)
10        # await swriter.awrite(s)
11        uart.write(s)
12        print(s)
13        await asyncio.sleep(2)

...
Silicon Labs CP210x USB to UART Bridge (COM4) / UART_test + v

Recieved b'Hello uart241\n'
Hello uart43 83 N  

Hello uart44 note

Recieved b'Hello uart242\n'
Hello uart45 83 N  

Hello uart46 note

Recieved b'Hello uart243\n'
Hello uart47 83 N  

Hello uart48 note

Recieved b'Hello uart244\n'
Hello uart49 83 N  

Hello uart50 note

Recieved b'Hello uart245\n'
Hello uart51 83 N  

Hello uart52 note

Recieved b'Hello uart246\n'
Hello uart53 83 N  

Hello uart54 note

Recieved b'Hello uart247\n'
Hello uart55 83 N  

Hello uart56 note

Recieved b'Hello uart248\n'
Hello uart57 83 N  

Hello uart58 note

Recieved b'Hello uart249\n'
Hello uart59 83 N  

Hello uart60 note
```

PYMAKR ...

PROJECTS

UART_test

Silicon Labs CP210...

DEVICES

Silicon Labs CP210x...

Silicon Labs CP210x...

main.py 2 x
main.py > receiver > [e] reader
1 import uasyncio as asyncio
2 from machine import UART
3 uart = UART(2, 115200)
4
5 async def sender():
6 i=200
7 while True:
8 s = 'Hello uart' + str(i)
9 uart.write(s)
10 print(s)
11 await asyncio.sleep(0.01)
12 i=i+1

...
Silicon Labs CP210x USB to UART Bridge (COM7) / unknown

Hello uart241
Hello uart242
Hello uart243
Hello uart244
Hello uart245
Hello uart246
Hello uart247
Hello uart248
Hello uart249

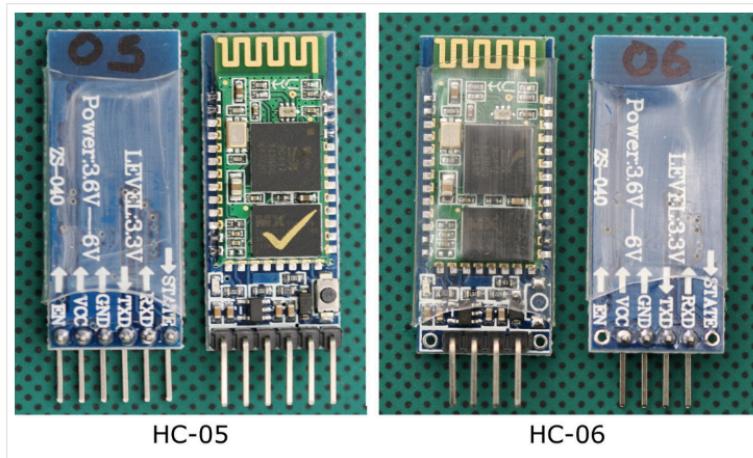
משימה 10 - תקשורת Bluetooth מבוססת HC-05 או HC-06

קישורים:

<https://www.martyncurrey.com/hc-05-and-hc-06-zs-040-bluetooth-modules-first-look/>

<https://heeed.net/micro-bit-and-the-blue-micropython/>

ההבדל בין HC-05 ל-HC-06



רכיב HC-05 יכול לשמש גם Master ו גם Slave בעודוhc-06 יכול לשמש רק Slave. במילים אחרות המשמעות היא שהיא-05 HC יכול ליצור חיבור לממשק אחר וה-06 HC יכול רק לקבל חיבור מממשק אחר. לאחר יצירת הקשר בשני הכוונים התקשרות יכולה להיות דו-כיוונית.

מאפיינים טכניים וחשמליים של הרכיב:

Radio Chip: CSR BC417

Memory: External 8Mbit Flash

Output Power: -4 to +6dbm Class 2

Sensitivity: -80dbm Typical

Bit Rate: EDR, up to 3Mbps

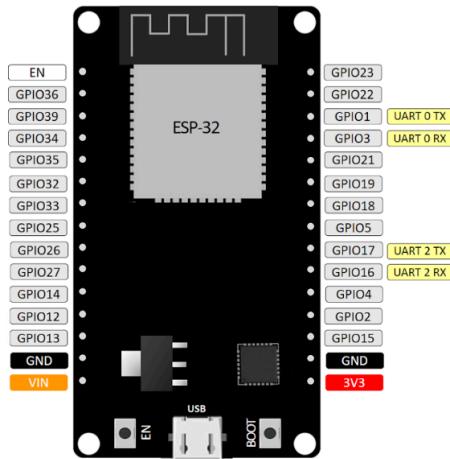
Interface: UART

Antenna: Built-in

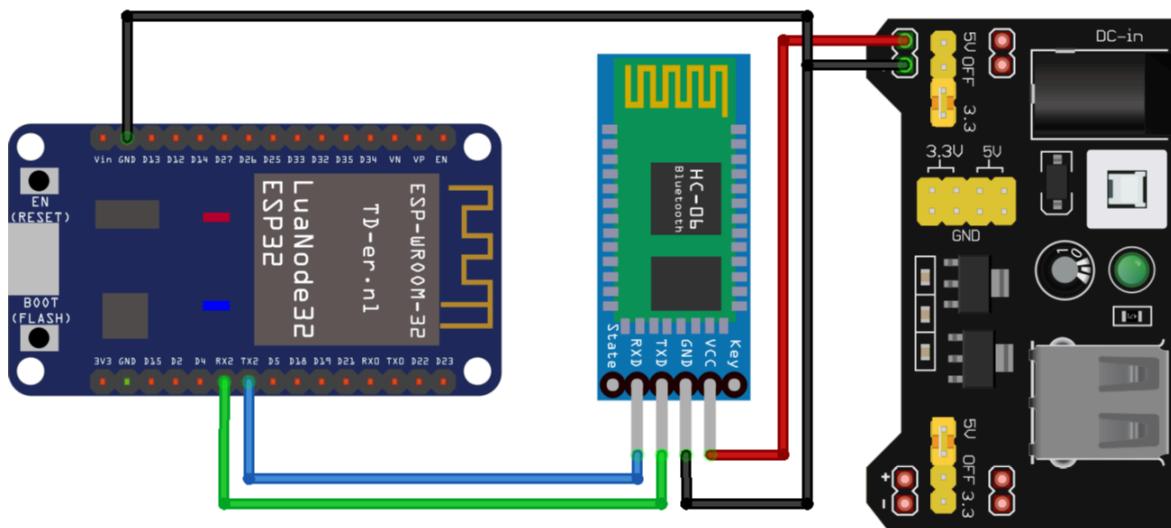
Voltage: 3.1 to 4.2VDC

Current: 40mA max

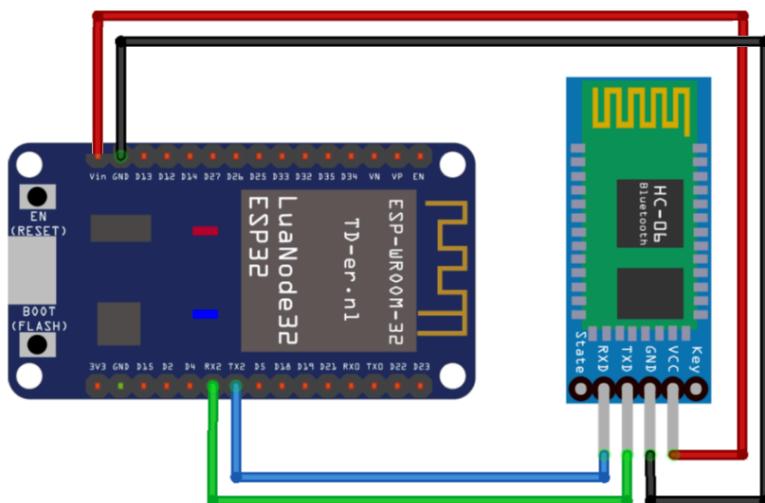
כפי שראינו במשימה הקודמת לבקר ESP32 יש 3 ממתקי תקשורת UART כאשר רק UART2 המתפרק בין הדקים (16 GPIO 17 and GPIO 20) זמין באופן מלא לשימוש. להלן מיקום הדק ה-UART בבלר:



נחבר את רכיב ה- BT ל-2 UART הדקים :rx=16 -I tx=17



אפשרות נוספת לחבר HC-05 או HC-06 מתואר באיזור הבא:



אתחול הגדרות ל- HC-05 (גרסת סינכרונית):

נכתב את הקוד הבא לבקר במטרה לאתחול את הגדרות התקשורת בין הטלפון הנייד לבין ה-HC-05. זאת במידה ואנו לא יודעים את הגדרות האתחול של הרכיב, כמו שם הרכיב, הסיסמה של הרכיב וקצב התקשורת שלו.

```
from machine import UART
import time

bt_uart = UART(2, baudrate=38400)
#bt_uart = UART(2, baudrate=9600)

print("Enter AT commands (type and press Enter):")

while True:
    cmd = input(">> ")
    bt_uart.write(cmd + "\r\n")

    time.sleep(0.2)

    while bt_uart.any():
        response = bt_uart.readline()
        if response:
            print(response.decode().strip())
```

אתחול הגדרות ל- HC-05 (גרסת אסינכרונית):

נכתב את הקוד הבא לבקר במטרה לאתחול את הגדרות התקשורת בין הטלפון הנייד לבין ה-HC-05. זאת במידה ואנו לא יודעים את הגדרות האתחול של הרכיב, כמו שם הרכיב, הסיסמה של הרכיב וקצב התקשורת שלו.

```
import uasyncio as asyncio
from machine import UART
import _thread

bt_uart = UART(2, baudrate=38400)
#bt_uart = UART(2, baudrate=9600)

async def bt_receiver():
    reader = asyncio.StreamReader(bt_uart)
    while True:
        try:
            line = await reader.readline()
            if line:
                print("[HC-05] >", line.decode().strip())
        except Exception as e:
            print("UART Read Error:", e)
            await asyncio.sleep(1)

def blocking_input_loop():
    print("Ready to send AT commands. Type and press Enter:")
    while True:
```

```

try:
    line = input("=> ")
    bt_uart.write(line + "\r\n")
except Exception as e:
    print("Input error:", e)

async def main():
    asyncio.create_task(bt_receiver())
    _thread.start_new_thread(blocking_input_loop, ())
    while True:
        await asyncio.sleep(1)

asyncio.run(main())

```

שימוש לב בשפת Python, הפעולה `input` היא סינכרונית וחוסמת (blocking), כלומר — כאשר הקוד מגע לשורה `input`, הוא נעצר לחילוטין וממתין לקלט מהמשתמש, ולא ניתן להמשיך להריצ' פעולות אחרות (כולל `async tasks`) בזמן המתנה.

הפתרון לזה יהיה שימוש ב- `thread` כך שהפעולה תופעל "כישום" נפרד ולא יחסום את שאר הקוד!
להלן דוגמת קוד המדגימה שימוש ב- `:thread`:

```

import _thread
import time

def th_func(delay, id):
    while True:
        time.sleep(delay)
        print('Running thread %d' % id)

_thread.start_new_thread(th_func, (1, 1))
_thread.start_new_thread(th_func, (2, 2))
_thread.start_new_thread(th_func, (3, 3))
_thread.start_new_thread(th_func, (2, 4))
_thread.start_new_thread(th_func, (1, 5))

```

ניתן לראות שלא מדובר בפעולת אסינכרונית ולמרות זאת היא מופעלת מספר פעמים במקביל.

```

Shell
Running thread 5
Running thread 2
Running thread 4
Running thread 1
Running thread 5
Running thread 3
Running thread 2
Running thread 4
Running thread 1
Running thread 5
Running thread 1

```

אתחול הגדרות ל- HC-06:

נכתב את הקוד הבא לבקר במטרה לאותחל את הגדרות התחברות בין הטלפון הנייד לבין ה-HC-06. זאת במידה ומם לא יודעים את הגדרות האתחול של הרכיב, כמו שם הרכיב, הסימנה של הרכיב וקצב התחברות שלו.

```
from machine import UART
import utime

uart = UART(2, 9600)
uart.init(9600, bits=8, parity=None, stop=1)
print(uart)

print("-----")
print("|\ Module HC-06 configuration |")
print("|\ enter AT           -- To test serial communication |")
print("|\ enter AT+NAME?????? -- To modify the module name |")
print("|\ enter AT+PIN1234   -- To modify the module PIN code |")
print("|\ enter AT+BAUD4     -- To modify the module communication speed|")
print("|\ Note: 1 for 1200,  2 for 2400,  3 for 4800,  4 for 9600 |")
print("|\      5 for 19200, 6 for 38400, 7 for 57600, 8 for 115200 |")
print("|\-----|")

while True:
    print("ENTER AT Commands: ")
    try:
        str = input()
        uart.write(str)
        utime.sleep_ms(100)
    except OSError:
        pass

    # wait for response
    start_time = utime.ticks_ms()
    timeout = False
    while not uart.any() and not timeout:
        if utime.ticks_diff(utime.ticks_ms(), start_time) > 500:
            timeout = True
    if timeout:
        print('Failed, response timed out')
    else:
        buf = uart.read()
        print("received:",buf)
    utime.sleep_ms(600)
```

להלן דוגמה לפט התוכנית:

```

;j32mI (2910) uart: ALREADY NULLm
UART(2, baudrate=9600, bits=8, parity=None, stop=1, tx=17, rx=16, rts=-1, cts=-1, txbuf=256, rxbuf=256, timeout=0, timeout_char=2)

| Module HC-06 configuration
| enter AT           -- To test serial communication
| enter AT+NAME?????  -- To modify the module name
| enter AT+PIN1234    -- To modify the module PIN code
| enter AT+BAUD4      -- To modify the module communication speed
| Note: 1 for 1200, 2 for 2400, 3 for 4800, 4 for 9600
|      5 for 19200, 6 for 38400, 7 for 57600, 8 for 115200

ENTER AT Commands:
AT
2
received: b'OK'
ENTER AT Commands:
AT+NAMEESP32_BT
15
received: b'OKsetname'
ENTER AT Commands:
AT+PIN1234
10
received: b'OKsetPIN'
ENTER AT Commands:
AT+BAUD4
8
received: b'OK9600'
ENTER AT Commands:

```

במידה והרכיב HC-06 אינו מגיב נקלט הודעה דומה זו:

```

Module HC-06 configuration
enter AT           -- To test serial communication
enter AT+NAME?????  -- To modify the module name
enter AT+PIN1234    -- To modify the module PIN code
enter AT+BAUD4      -- To modify the module communication speed
Note: 1 for 1200, 2 for 2400, 3 for 4800, 4 for 9600
      5 for 19200, 6 for 38400, 7 for 57600, 8 for 115200

ENTER AT Commands:
AT
2
Failed, response timed out
ENTER AT Commands:
AT
2
received: b'OK'
ENTER AT Commands:

```

להלן קוד תוכנית נוסף לאתחול רכיב HC-06

```

from machine import UART
import utime

"""
ESP32 UART2          HC-06 / CH-05
GPIO_17_UART2_TX     RX
GPIO_16_UART2_RX     TX

To enter AT-Command mode in HC05:
Press & Hold the onboard button while power on.

To enter AT-Command mode in HC06:
Power-up in NOT CONNECTED

Baudrate for at-command mode in HC05: 38400
Baudrate for at-command mode in HC06: 9600

```

```

"""
NAME = "MGKBluetooth4"
PASSWORD = "1234"

uart2 = UART(2,baudrate=9600)      # at-command baudrate for HC06
print(uart2)

#2 sec timeout is arbitrarily chosen
def sendAT(cmd, uart=uart2, timeout=2000):
    print("CMD: " + cmd)
    uart.write(cmd)
    waitResp(uart, timeout)

def waitResp(uart=uart2, timeout=2000):
    prvMills = utime.ticks_ms()
    resp = b""
    while (utime.ticks_ms() - prvMills) < timeout:
        if uart.any():
            resp = b"".join([resp, uart.read(1)])
    decoded_string = resp.decode("utf-8")
    print(decoded_string)

#commands for HC-05 version:  VERSION:3.0-20170609
print("---- Start ----")
waitResp()
sendAT("AT\r\n")
sendAT("AT+ORGL\r\n")           #Restore default setting
sendAT("AT+VERSION\r\n")
sendAT("AT+UART?\r\n")
sendAT("AT+UART=9600,0,0\r\n")  #9600 baud, 1 stop, parity=none
sendAT("AT+UART?\r\n")
sendAT("AT+PSWD?\r\n")
sendAT("AT+PSWD=\\""+PASSWORD+"\r\n")  #Set PIN = "1234"
sendAT("AT+PSWD?\r\n")
sendAT("AT+NAME=\\""+NAME+"\r\n")
sendAT("AT+NAME?\r\n")
sendAT("AT+ADDR?\r\n")
print("---- Done ----")

#commands for HC-06 version:  hc01.comV2.0 , linvorV1.8
# print("---- Start ----")
# waitResp()
# sendAT("AT")
# sendAT("AT+VERSION")
# sendAT("AT+BAUD4")          #4 --> 9600
# sendAT("AT+NAME"+NAME)
# sendAT("AT+PIN"+PASSWORD)
# sendAT("AT+PN")             #AT+PN sets no parity
# print("---- Done ----")

```

לאחר אתחול הרכיב נועבור לתוכנית המבוצעת תקשורת טקוט דו כיוון בין הבקר לטלפון נייד

קוד התוכנית:

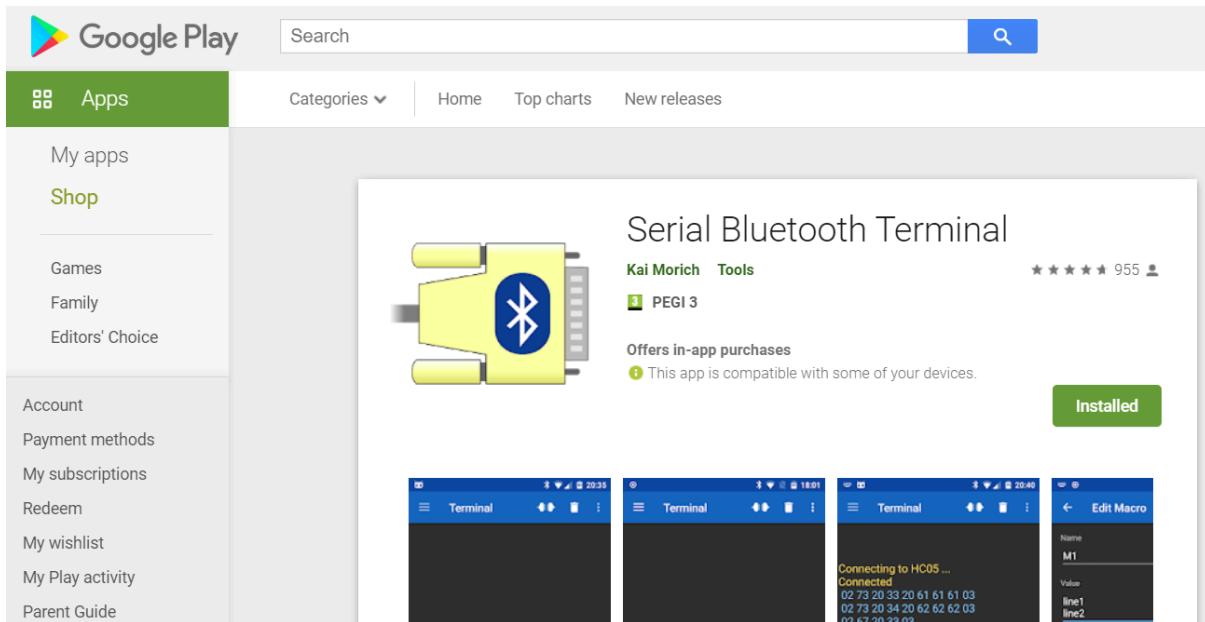
```
from machine import UART
import utime

uart = UART(2, 9600)
uart.init(9600, bits=8, parity=None, stop=1)
print(uart)

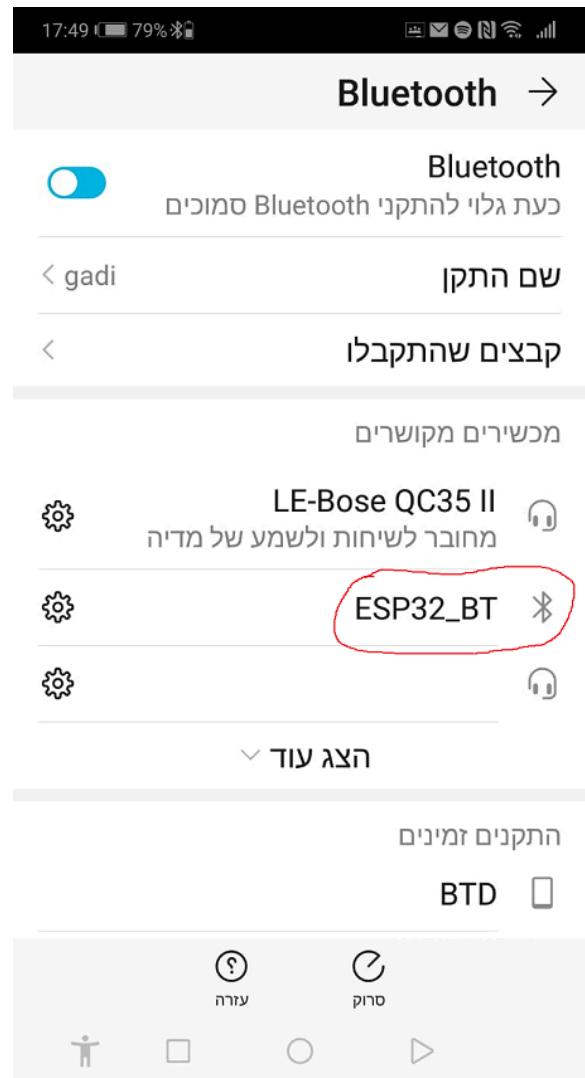
while True:
    if uart.any():
        while uart.any():
            buf = uart.read()
            print('received:',buf)
            utime.sleep_ms(15)

        utime.sleep_ms(10)
        try:
            uart.write("OK")
            print('sent response')
        except OSError:
            pass
```

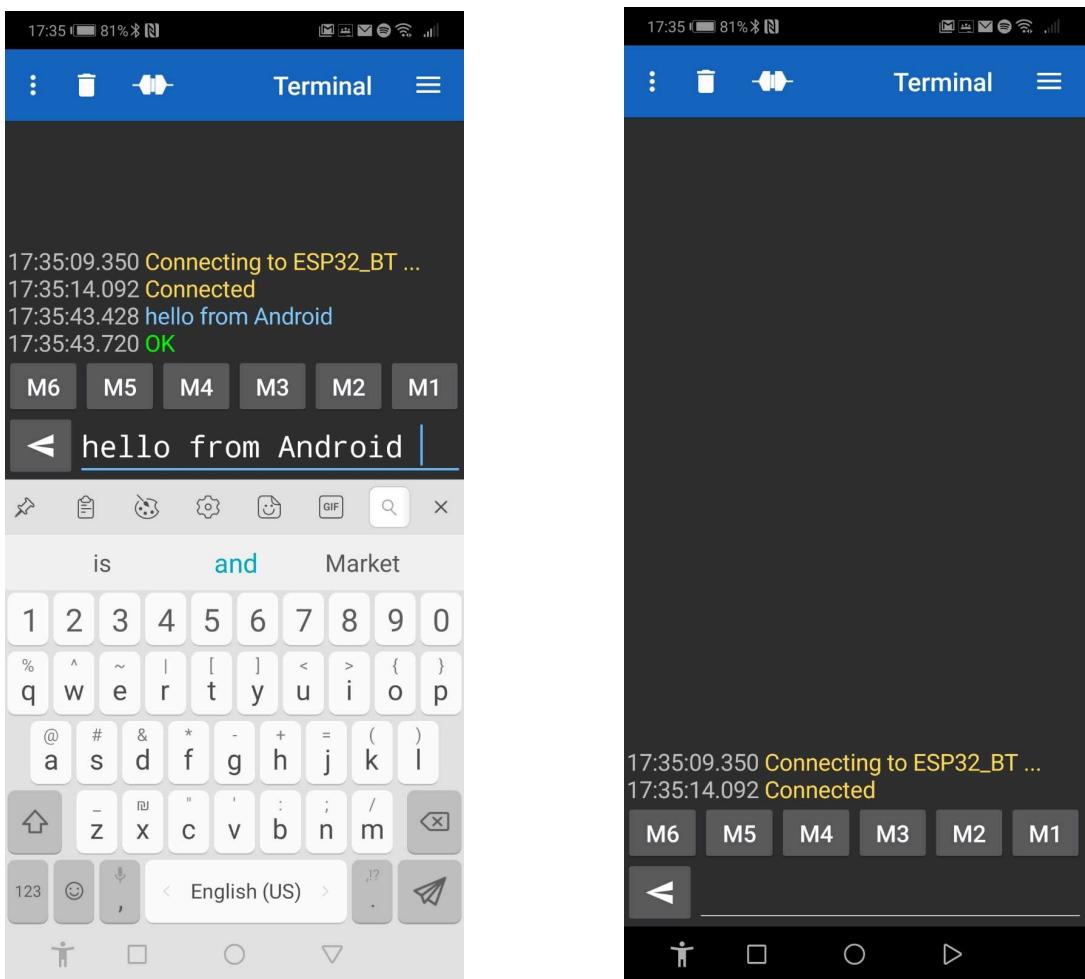
נפתח תוכנה ייעודית לתקשורת BT דרך הטלפון הנייד, כמו התוכנה הבאה:



נחבר את הטלפון נייד למכשיר BT על ידי כף שנסורק את התקני ה-Bluetooth הזמינים:



לבסוף נפעיל את היישום שהתקנו ונשלח לבקר טקסט:



על מסך המחשב נקלט הפלט הבא:

```

import time
import utime
|
uart = UART(2, 9600)
uart.init(9600, bits=8, parity=None, stop=1)
print(uart)

while True:
    if uart.any():
        while uart.any():
            buf = uart.read()
            print('received:',buf)
            utime.sleep_ms(15)

        utime.sleep_ms(10)
        try:
            uart.write("OK")
            print('sent response')
        except OSError:
            pass

```

On the right side of the code editor, there is a vertical scroll bar and a list of line numbers from 605 to 634. A red oval highlights the text output area, which contains the following:

```

605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
sent response
received: b'hello from Android \r\n'
sent response

```

להלן דוגמת קוד נוספת לתקשרות דו-כיוונית בין בקר ESP32 לבין טלפון נייד:

```
from machine import UART
import asyncio
from random import randint

uart = UART(2, 9600)
uart.init(9600, bits=8, parity=None, stop=1)
print(uart)

async def myTask1(lock):
    while True:
        try:
            await lock.acquire()
            if uart.any():
                data = uart.readline()
                #print('received:',data)
                # Convert byte string to a string using the decode() method
                decoded_string = data.decode("utf-8")
                print('Data: ', decoded_string , type(decoded_string))

        except asyncio.CancelledError:
            print("Peripheral task cancelled")
        except Exception as e:
            print("Error in ConnectionTask:", e)
        finally:
            await asyncio.sleep_ms(15)
            lock.release()

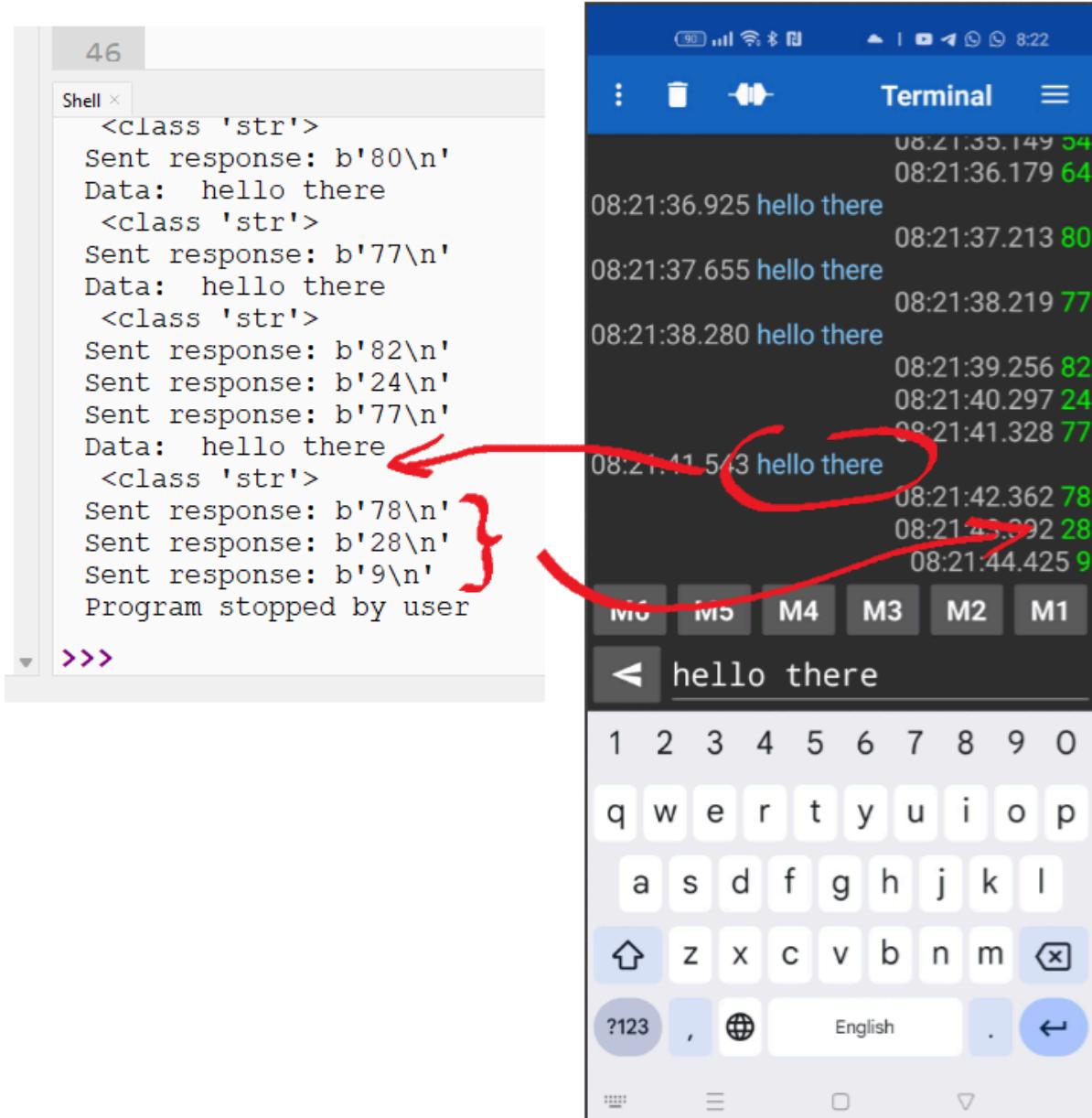
async def myTask2(lock):
    while True:
        try:
            await lock.acquire()
            num = randint(0,100)
            Data = str(num)+"\n"
            # Convert string to byte string using the encode() method
            sendData = Data.encode('utf-8')
            uart.write(sendData)
            print('Sent response:', sendData)
        except asyncio.CancelledError:
            print("Peripheral task cancelled")
        except Exception as e:
            print("Error in ConnectionTask:", e)
        finally:
            await asyncio.sleep_ms(1000)
            lock.release()

#Run all tasks at the same time
async def main():
    lock = asyncio.Lock() # Main Lock instance
    t1 = asyncio.create_task(myTask1(lock))
    t2 = asyncio.create_task(myTask2(lock))
    await asyncio.gather(t1, t2)
```

```
#Running the main program
try:
    asyncio.run(main())
except KeyboardInterrupt:
    print("Program stopped by user")
```

שימוש לב! קוד זה עושה שימוש בתכנות אסינכרוני, מומלץ לגשת לנוסף ב' של ספר זה כדי ללמידה יותר על תכנות אסינכרוני.

נקבל את הפלט הבא:



משימה 11 - תקשורת Bluetooth Low Energy בברker ESP32

קישורים:

<https://randomnerdtutorials.com/micropython-esp32-bluetooth-low-energy-ble/>

<https://docs.micropython.org/en/latest/library/bluetooth.html>

Bluetooth Low Energy היא טכנולוגיה להעברת מידע אלחוטית בין התקנים. לעומת זאת, Bluetooth היא גרסה לצריכת חשמל נמוכה המיועדת להתקנים קטנים שצריכים חמי סוללה ארוכים, כמו צמידי כושר ושעונים חכמים.

- Bluetooth: מתאים להעברת מידע מהיר, כמו העברת שמע לאוזניות או רמקולים. פרוטוקול זה תומך בהעברה בקצבים גבוהים, אך דורש צריכת חשמל גדולה יחסית.
- Bluetooth Low Energy: פועל בצריכת חשמל נמוכה ויכול לפעול זמן רב. מתאים להתקנים קטנים שצריכים חמי סוללה ארוכים אך סובל מקצב העברת מידע מוגבל. טכנולוגיה זו שימושית בעיקר לישומי IoT.

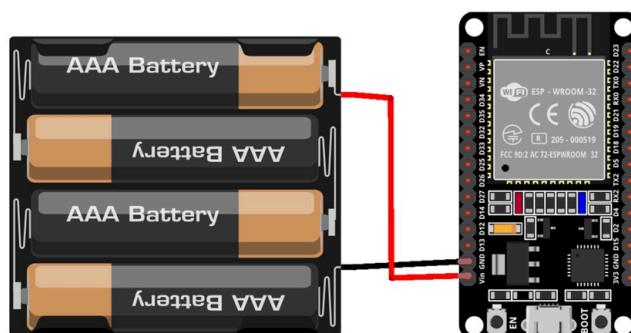
הצורף BLE:

1. צריכת אנרגיה נמוכה - מתאים למכשירים המופעלים על סוללה
2. תקשורת אלחוטית לטווח קצר בין מכשירים
3. תמייה נרחבת במכשירים ניידים ומערכות הפעלה מודרניות
4. אידיאלי לשילוח כמפורטות קטנות של מידע באופן תקופתי

ברker ESP32 כולל רכיב Bluetooth Low Energy מובנה שלו נטמון בפעריות זו

חשיבות: שימוש במקור מתח חיצוני

העובדת עם רכיב Bluetooth Low-level או רכיב Wi-Fi המובנים בברker ESP32 דורש זרם עבודה גדול מ-500mA המספק מחברת ה-USB. על כן יש צורך לחבר מקור מתח חיצוני כמתואר באירור



הדק Vin ב-ESP32 מחובר למיציב מתח פנימי. על כן ניתן לבדוק בהדק Vin מתח בין 5V ל-12V. כל מתח בטוח זהה בהדק Vin עובר למיציב המפתח את אותו ל-3.3V ולאחר מכן מזון לצירוף ההיקפי של לוח ESP32.

באמצעות סוללה חיצונית של 6V או 7V יכול להפעיל את ESP32 דרך פין Vin על ידי חיבור GND של ESP32 עם GND של הסוללה. ניתן לחבר כל מתח בין 5V ל-12V לפין Vin ESP32 אולם מומלץ לא להשתמש ביותר מסוללה חיצונית של 7V. מכיוון ש-ESP32 צריך רק 3.3V כדי לפעול, שאר המתחים מופזרים על ידי מיציב המתח חום.

ניסוי 1 - תקשורת BLE תוך שימוש בספרייה ubluetooth

מקור:

<https://docs.micropython.org/en/latest/library/bluetooth.html>

<https://www.youtube.com/watch?v=RvbWl8rZOoQ>

דוגמת קוד ללא שימוש במחלקה

להלן הקוד:

```
import ubluetooth
import utime
import urandom
from machine import Timer

ble = ubluetooth.BLE()
ble.active(True)

#Personal UUID generator https://www.uuidgenerator.net
UART_UUID = ubluetooth.UUID("6E400001-B5A3-F393-E0A9-E50E24DCCA9E")
TX_UUID = ubluetooth.UUID("6E400003-B5A3-F393-E0A9-E50E24DCCA9E")
RX_UUID = ubluetooth.UUID("6E400002-B5A3-F393-E0A9-E50E24DCCA9E")

UART_SERVICE = (
    UART_UUID,
    (
        (TX_UUID, ubluetooth.FLAG_NOTIFY),
        (RX_UUID, ubluetooth.FLAG_WRITE),
    ),
)
)

handles = ble.gatts_register_services((UART_SERVICE,))
tx_handle = handles[0][0]
rx_handle = handles[0][1]
conn_handle = None

def advertise():
    name = b'ESP32-BLE'
    adv_data = bytearray()
    adv_data += bytes((len(name) + 1, 0x09)) + name
    ble.gap_advertise(100, adv_data)

def bt_irq(event, data):
    global conn_handle
    if event == 1:
        conn_handle = data[0]
        print("Connected")
    elif event == 2:
        print("Disconnected")
        conn_handle = None
        advertise()
    elif event == 3:
        msg = ble.gatts_read(rx_handle).decode('utf-8').strip()
```

```

print("Received:", msg)

ble.irq(bt_irq)
advertise()

def send_random(timer):
    if conn_handle is not None:
        value = str(urandom.getrandbits(8))
        print("Sending:", value)
        ble.gatts_notify(conn_handle, tx_handle, value)

timer = Timer(0)
timer.init(period=5000, mode=Timer.PERIODIC, callback=send_random)
print("Waiting to connect...")

```

כדי להתחבר ל-ESP32 בתצורת BLE, השתמשו באפליקציה שנועדה לכך כדוגמת nRF Connect for Mobile (חינמית):

Android: <https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp>

iOS: <https://apps.apple.com/us/app/nrf-connect-for-mobile/id1054362403>

AIR להתחבר עם:nRF Connect

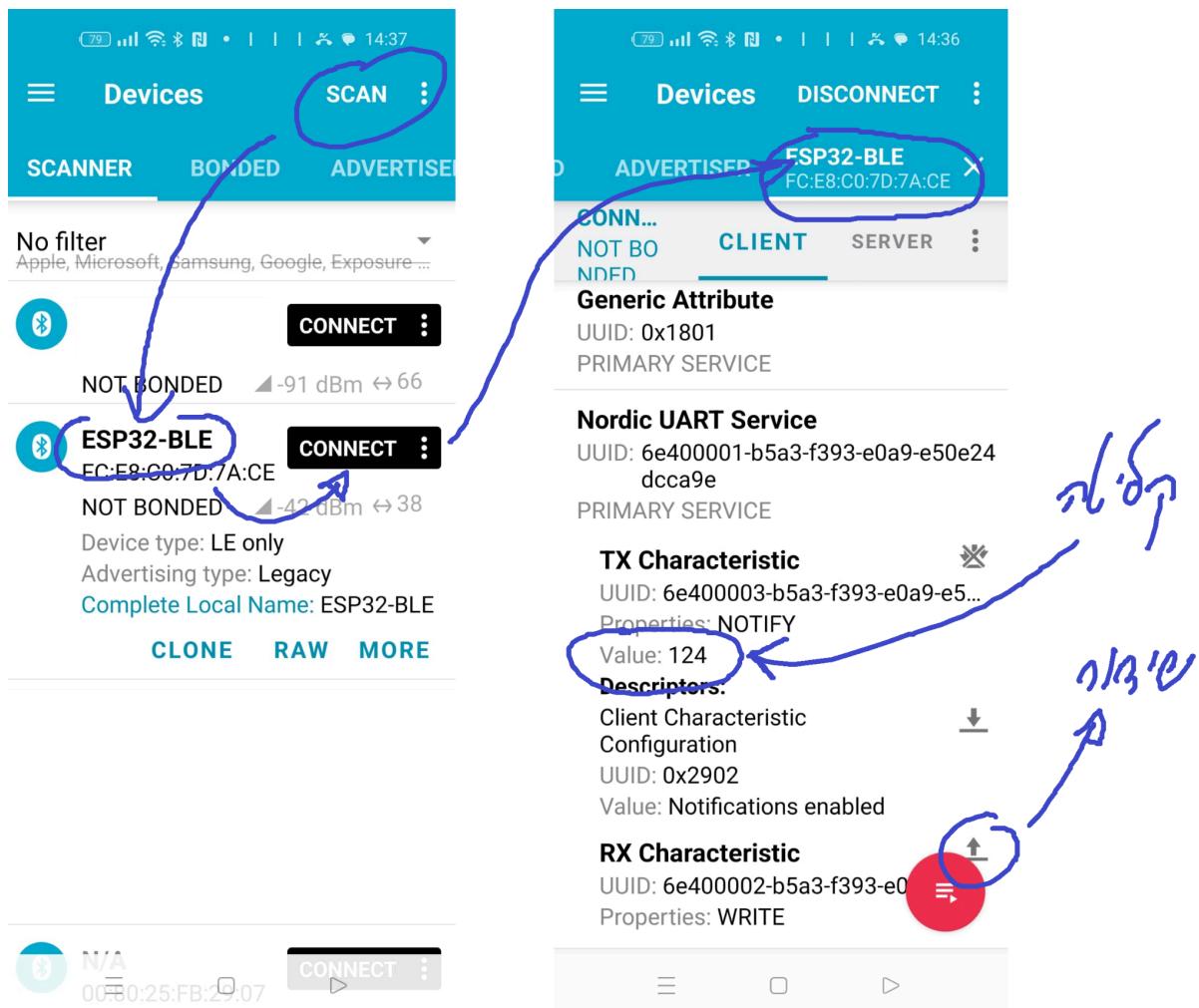
- הפעילו את הקוד על ה-ESP32.
- וDAO שההתקן משליך (הוא אמר להופיע בשם ESP32-BLE).
- פתחו את האפליקציה nRF Connect.
- סורקנו את הסביבה – אמר להופיע ההתקן שלך בשם ESP32-BLE.
- לחזו על Connect.
- תראו שירות עם SID שונראה כך:

6E400001-B5A3-F393-E0A9-E50E24DCCA9E

הוא השירות של UART.

- תראו שני מאפיינים (characteristics):
- אחד עם Write – כדי לשלווח טקסט ל-ESP32.
- אחד עם Notify – כאן יופיעו המספרים האקריאים שאתם משלדרים.
- לחזו על Write ותשלווח טקסט – תראו את הודעה מופיעה במסך של ה-ESP32 (ב-print).
- כל 5 שניות תראו באפליקציה הודעה חדשה – המספר האקריא שהbakr משליך.

להלן דוגמה לאופן השימוש במסדי היישום:



להלן גרסה משופרת של הקוד העושה שימוש במחלקה ובכך מקצר את הקוד הראשי של התוכנית:

```
import ubluetooth
import urandom
from machine import Timer

class BLEUART:
    def __init__(self, name="ESP32-BLE"):
        self.ble = ubluetooth.BLE()
        self.ble.active(True)
        self.ble.irq(self.bt_irq)

        self.tx = None
        self.rx = None
        self.conn_handle = None

        # UART Service UUID
        UART_UUID = ubluetooth.UUID("6E400001-B5A3-F393-E0A9-E50E24DCCA9E")
        # RX Characteristic (write)
        RX_UUID = ubluetooth.UUID("6E400002-B5A3-F393-E0A9-E50E24DCCA9E")
```

```

# TX Characteristic (notify)
TX_UUID = ubluetooth.UUID("6E400003-B5A3-F393-E0A9-E50E24DCCA9E")

self.tx = (TX_UUID, ubluetooth.FLAG_NOTIFY)
self.rx = (RX_UUID, ubluetooth.FLAG_WRITE)

UART_SERVICE = (UART_UUID, (self.tx, self.rx))
SERVICES = (UART_SERVICE, )

((self.tx_handle, self.rx_handle), ) =
self.ble.gatts_register_services(SERVICES)
self.ble.gap_advertise(100, adv_data=self._advertise(name))

def _advertise(self, name):
    name_bytes = bytes(name, 'utf-8')
    adv_data = bytearray()
    adv_data += bytes((len(name_bytes) + 1, 0x09)) + name_bytes
    return adv_data

def bt_irq(self, event, data):
    if event == 1: # central connected
        self.conn_handle, _, _ = data
        print("Device connected")
    elif event == 2: # central disconnected
        print("Device disconnected")
        self.conn_handle = None
        self.ble.gap_advertise(100,
adv_data=self._advertise("ESP32-BLE"))
    elif event == 3: # write to RX
        conn_handle, attr_handle = data
        msg =
self.ble.gatts_read(self.rx_handle).decode('utf-8').strip()
        print("Received:", msg)

def send(self, data):
    if self.conn_handle is not None:
        self.ble.gatts_notify(self.conn_handle, self.tx_handle, data)

ble_uart = BLEUART()

def send_random(t):
    num = str(urandom.getrandbits(8)) # 255-ל-0
    print("Sending:", num)
    ble_uart.send(num)

# שליחה כל 5 שניות
timer = Timer(0)
timer.init(period=5000, mode=Timer.PERIODIC, callback=send_random)

```

ניסוי 2 - תקשורת BLE תוך שימוש בספרייה aiobile

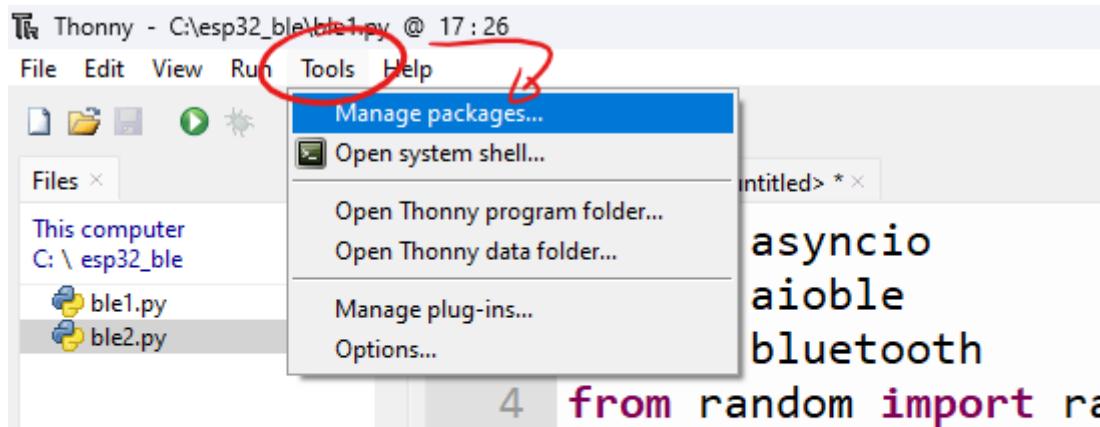
התקנת ספרייה "יעודית"

בהתאם להמלצת מפתחי MicroPython אנו עושים שימוש בספרייה "יעודית" בשם aioble לצורכי יצירת תקשורת בין מיקרו בקר ESP32 לבין טלפון נייד. להלן מספר פעולות עיקריות ב-aioble:

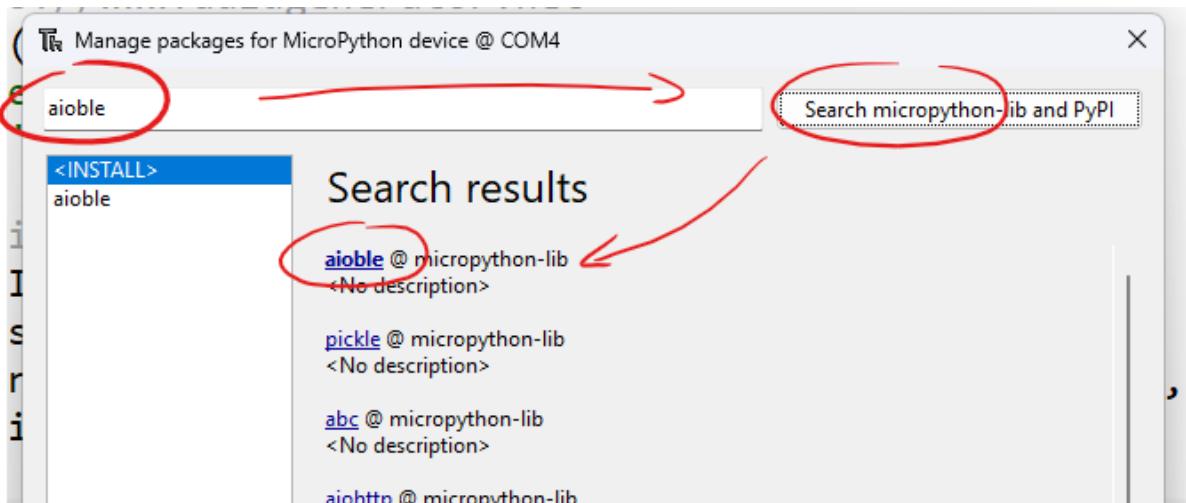
aioble.Service	מגדלר שירות BLE המכיל מספר מאפיינים (Characteristics)
aioble.Characteristic	מגדלר מאפיין בתוך שירות שיכל להכיל מידע
aioble.Peripheral	מייצג התקן BLE פריפריאלי (שרת)
aioble.Central	מייצג התקן BLE מרכזי (לקוח)
aioble.scan	סורך אחר מכשירי BLE בסביבה

שימוש לב! את החבילת אנו מתקינים ישירות על הבקר, על כן יש לחבר אותו למחשב לפני תחילת תהליך ההתקנה.

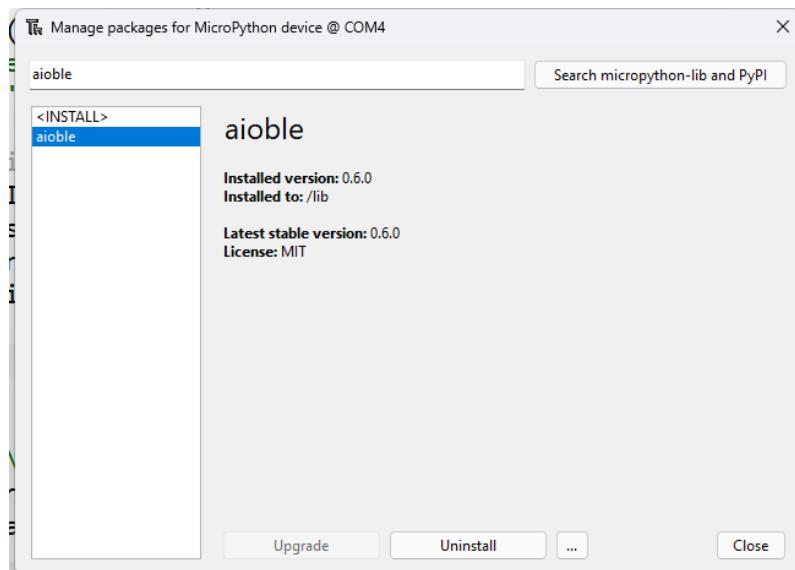
כדי להתקין את חבילת הקוד aioble נפתח את סביבת העבודה Thonny ונלחץ על tools→Manage packages



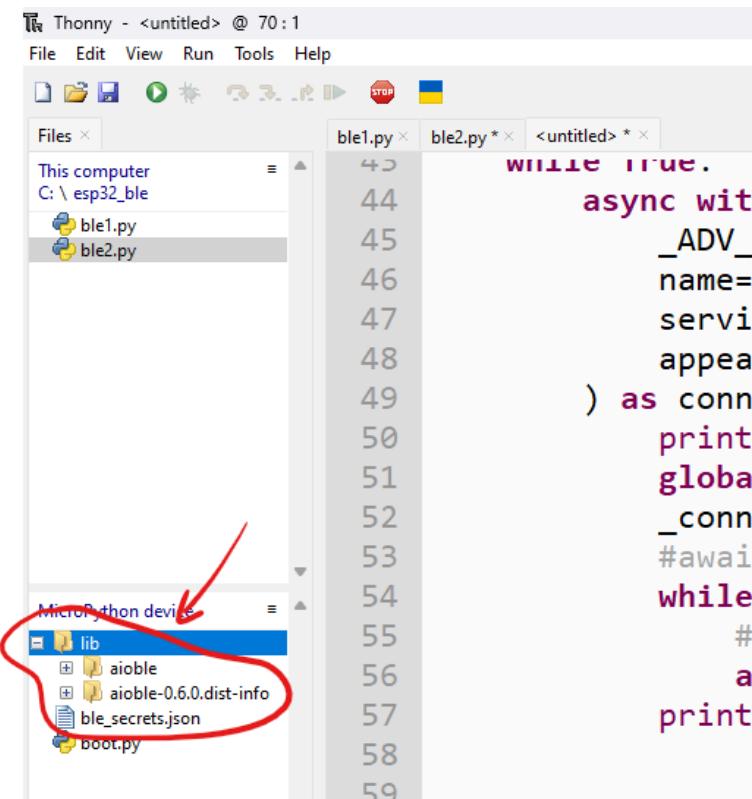
נחפש את החבילת aioble



ואז נתקין אותה



שיםו לב שלאחר ההתקנה תפתח בברך ספרייה חדשה בשם lib הכוללת את הקבצים של חבילת התוכנה:



להלן דוגמת קוד לתקשרות דו-כיוונית בין בקר ESP32 לבין טלפון נייד:

```

import asyncio
import aioble
import bluetooth
from random import randint

#Personal UUID generator https://www.uuidgenerator.net
SERVICE_UUID = bluetooth.UUID('2b363f24-351f-4640-80ed-cb1f210228aa')
SEND_UUID = bluetooth.UUID('9ecdd7ad-48ad-40f2-af97-4872d2d90324')
RECEIV_UUID = bluetooth.UUID('f8d16e04-1304-4b43-8e4d-189bee24ab7a')

#create service and characteristics
service = aioble.Service(SERVICE_UUID)
sendChara = aioble.Characteristic(service, SEND_UUID, read=True,
notify=True)
receivChara = aioble.Characteristic(service, RECEIV_UUID, read=True,
write=True, notify=True, capture=True)
aioble.register_services(service)

async def sendDataTask():
    while True:
        num = randint(0,100)
        sendData = str(num)+"\n"
        # Convert string to byte string using the encode() method
        sendData = sendData.encode('utf-8')
        sendChara.write(sendData, send_update=True)
        print('Send data: ', sendData)

```

```

        await asyncio.sleep_ms(2000)

async def ConnectionTask():
    while True:
        try:
            con = await aioble.advertise(250_000, name="ESP32_BLE",
services=[SERVICE_UUID])
            print("Connection from", con.device)
            await con.disconnected()
        except asyncio.CancelledError:
            print("Peripheral task cancelled")
        except Exception as e:
            print("Error in ConnectionTask:", e)
    finally:
        await asyncio.sleep_ms(200)

async def ReceivingTask():
    while True:
        try:
            connection, data = await receivChara.written()
            print('Data: ', data , type(data))
            # Convert byte string to a string using the decode() method
            decoded_string = data.decode("utf-8")
            print('Data: ', decoded_string , type(decoded_string))
        except asyncio.CancelledError:
            print("Peripheral task cancelled")
        except Exception as e:
            print("Error in ReceivingTask:", e)
    finally:
        await asyncio.sleep_ms(100)

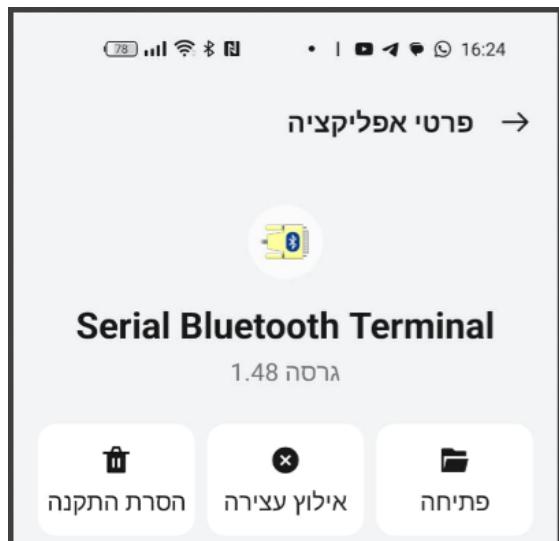
#Run all tasks at the same time
async def main():
    t1 = asyncio.create_task(ConnectionTask())
    t2 = asyncio.create_task(sendDataTask())
    t3 = asyncio.create_task(ReceivingTask())
    await asyncio.gather(t1, t2, t3)

#Running the main program
try:
    asyncio.run(main())
except KeyboardInterrupt:
    print("Program stopped by user")

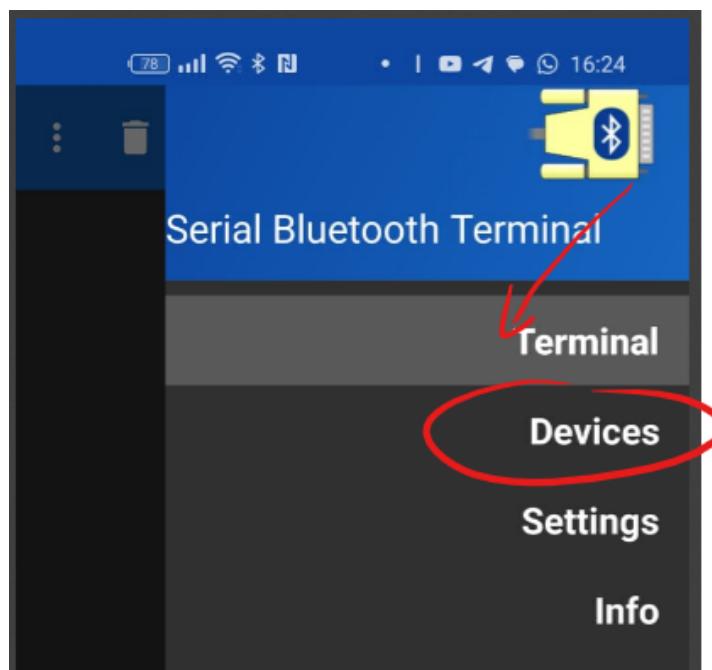
```

קטע קוד זה יכול שימוש בתכנות אסינכרוני. על כן מומלץ לעבור לנספח ב' וללמוד עקרונות תכנות אסינכרוני.

כדי לבדוק את התוכנה נפעיל את האפליקציה Serial Bluetooth Terminal

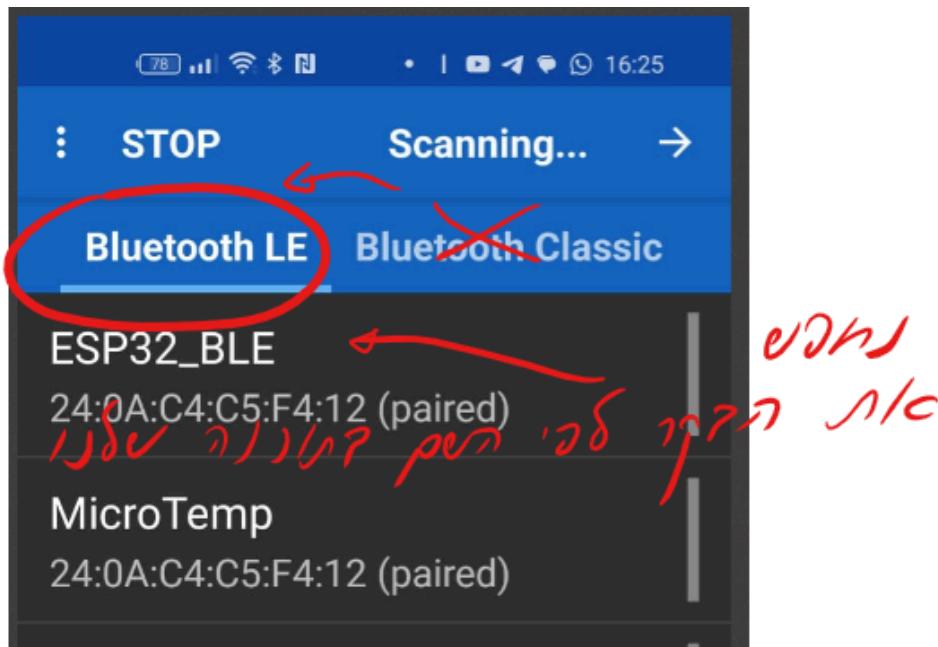


לאחר הפתיחה נלחץ כל Devices:

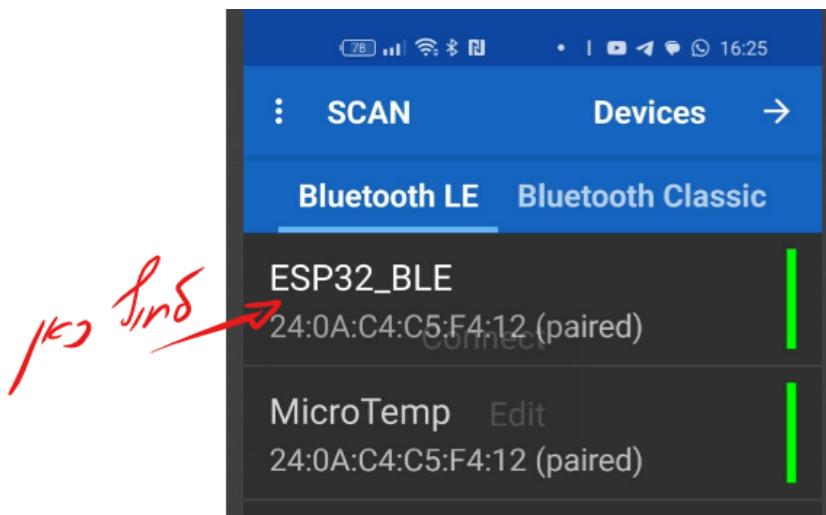


נלחץ על Scan ונחפש את שם הבקר כפי שרשמנו אותו בקוד התוכנה (כאן):

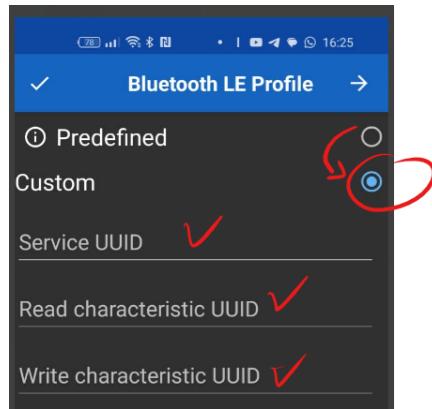
```
async def ConnectionTask():
    while True:
        try:
            con = await aioble.advertise(250_000, name="ESP32_BLE",
services=[SERVICE_UUID])
            print("Connection from", con.device)
            await con.disconnected()
```



נלחץ לחיצה ארוכה כל שנו של התחן כדי להיכנס להגדרותה שלו.



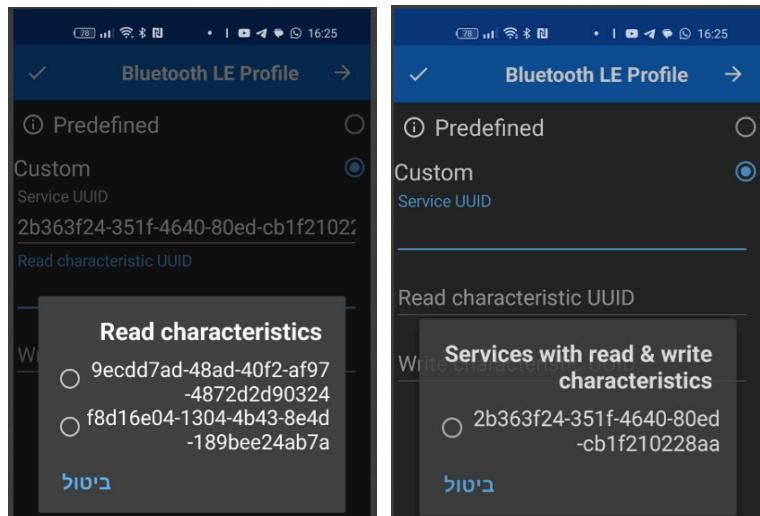
נקבל את החלון הבא:



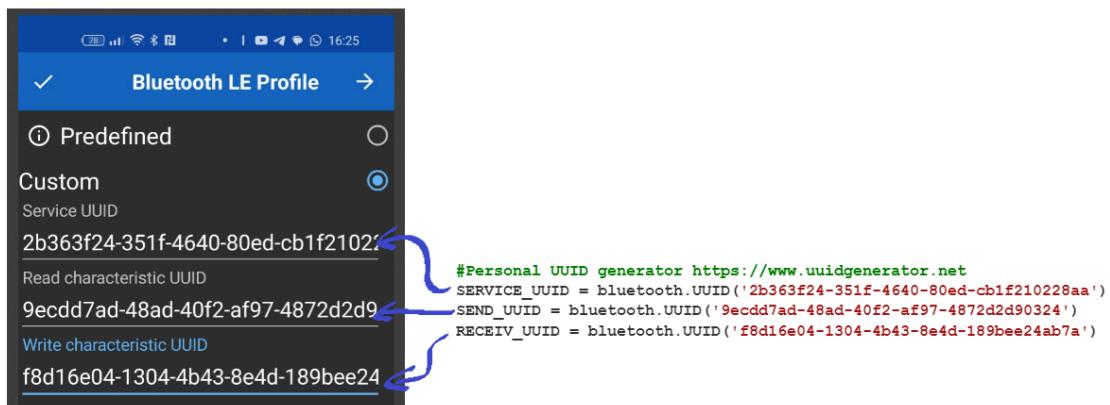
בחלון זו נשלים את הקוד ה-UUID כפי שרשمنו אותו בקוד התוכנה (כאי..)

```
#Personal UUID generator https://www.uuidgenerator.net
SERVICE_UUID = bluetooth.UUID('2b363f24-351f-4640-80ed-cb1f210228aa')
SEND_UUID = bluetooth.UUID('9ecdd7ad-48ad-40f2-af97-4872d2d90324')
RECEIV_UUID = bluetooth.UUID('f8d16e04-1304-4b43-8e4d-189bee24ab7a')
```

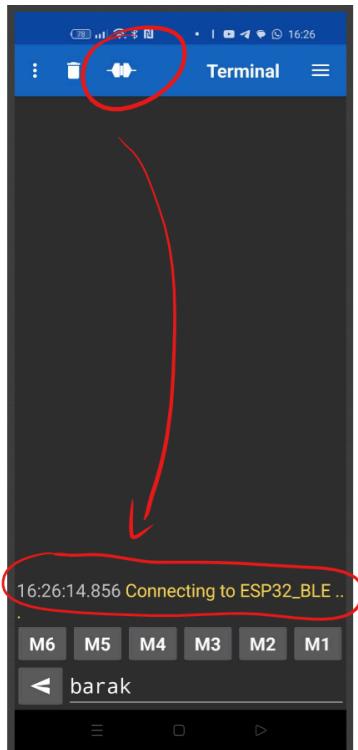
באופן הבא:



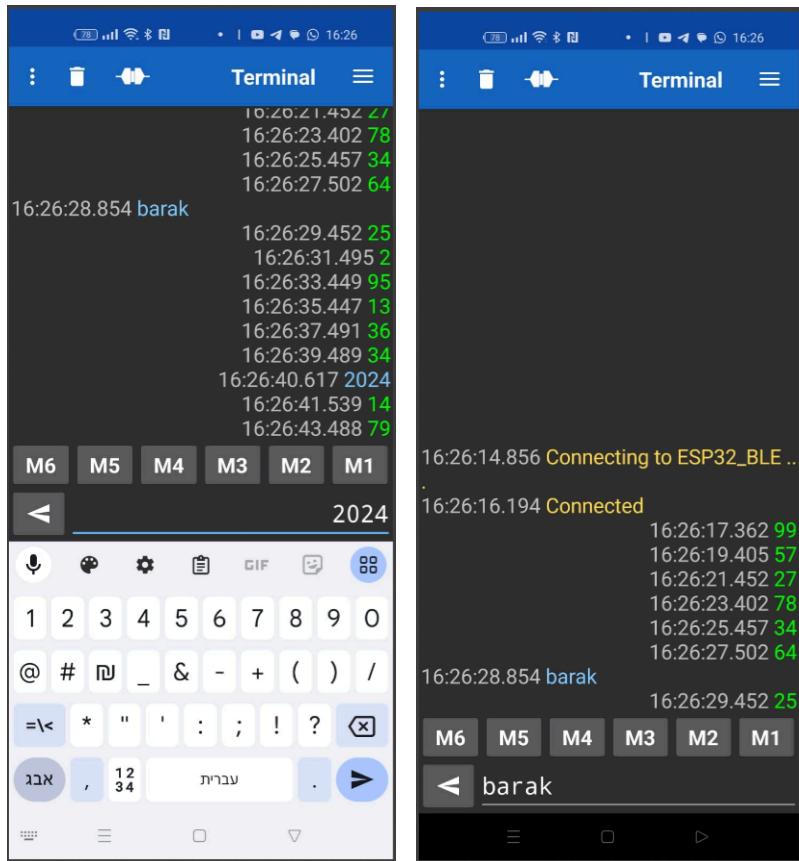
ואז קיבל את התוצאה הבא:



לאחר ההתחברות ניתן יהיה לשלוח ולקבל נתונים



להלן דוגמה:



בצד הבקר נקלט את הפלט הבא:

```

Shell >
Send data: b'11\n'
Send data: b'64\n'
Send data: b'80\n'
Send data: b'36\n'
Connection from Device (ADDR_RANDOM, 5a:28:0b:7e:64:85, CONNECTED)
Send data: b'99\n'
Send data: b'57\n'
Send data: b'27\n'
Send data: b'78\n'
Send data: b'34\n'
Send data: b'64\n'
Data: b'barak\r\n' <class 'bytes'>
Data: barak
<class 'str'>
Send data: b'25\n'
Send data: b'2\n'
Send data: b'95\n'
Send data: b'13\n'
Send data: b'36\n'
Send data: b'34\n'
Data: b'2024\r\n' <class 'bytes'>
Data: 2024
<class 'str'>
Send data: b'14\n'
Send data: b'79\n'

```

להלן דוגמה מעשית:

חיבור חיישן LM35 דרך כניסה ADC (אנלוגית) ושידור הטמפרטורה במקום שידור מספר אקראי כמו בדוגמה הקודמת.

בדיקת מילת מפתח ("OFF" / "ON") להפעלת או חיבור LED.

להלן הקוד:

```

import asyncio
import aioble
import bluetooth
from machine import Pin, ADC
import time

SERVICE_UUID = bluetooth.UUID('2b363f24-351f-4640-80ed-cb1f210228aa')
SEND_UUID = bluetooth.UUID('9ecdd7ad-48ad-40f2-af97-4872d2d90324')
RECEIV_UUID = bluetooth.UUID('f8d16e04-1304-4b43-8e4d-189bee24ab7a')

service = aioble.Service(SERVICE_UUID)
sendChara = aioble.Characteristic(service, SEND_UUID, read=True,
                                    notify=True)
receivChara = aioble.Characteristic(service, RECEIV_UUID, read=True,
                                      write=True, notify=True, capture=True)
aioble.register_services(service)

led = Pin(2, Pin.OUT) # LED
temp_sensor = ADC(Pin(34)) # LM35
temp_sensor.atten(ADC.ATTN_11DB)

def read_temperature():
    voltage = temp_sensor.read() * 3.3 / 4095
    temp_c = voltage * 100

```

```

        return temp_c

async def sendDataTask():
    while True:
        temperature = read_temperature()
        sendData = f"{temperature:.2f} C\n".encode('utf-8')
        sendChara.write(sendData, send_update=True)
        print('Send temperature:', sendData)
        await asyncio.sleep_ms(2000)

async def ConnectionTask():
    while True:
        try:
            con = await aioble.advertise(250_000, name="ESP32_BLE",
services=[SERVICE_UUID])
            print("Connection from", con.device)
            await con.disconnected()
        except asyncio.CancelledError:
            print("Peripheral task cancelled")
        except Exception as e:
            print("Error in ConnectionTask:", e)
        finally:
            await asyncio.sleep_ms(200)

async def ReceivingTask():
    while True:
        try:
            connection, data = await receivChara.written()
            decoded = data.decode("utf-8").strip().upper()
            print("Received command:", decoded)
            if decoded == "ON":
                led.on()
                print("LED turned ON")
            elif decoded == "OFF":
                led.off()
                print("LED turned OFF")
            else:
                print("Unknown command")
        except asyncio.CancelledError:
            print("Peripheral task cancelled")
        except Exception as e:
            print("Error in ReceivingTask:", e)
        finally:
            await asyncio.sleep_ms(100)

async def main():
    t1 = asyncio.create_task(ConnectionTask())
    t2 = asyncio.create_task(sendDataTask())
    t3 = asyncio.create_task(ReceivingTask())
    await asyncio.gather(t1, t2, t3)

try:
    asyncio.run(main())
except KeyboardInterrupt:

```

```
print("Program stopped by user")
```

משימה 12 - אתחול קישוריות ה-WiFi בברker ESP32

ברker ESP32 מספק ביצועים גבוהים על בסיס מעבד הכלול 2 ליבוט. כמו כן הברker כולל קישוריות ישירה לרשת האינטרנט דרך רכיב WiFi מובנה. כמו כן בהמשך משימה זו נעשה היכרות על הקבצים `uf2` ו- `main.py`.

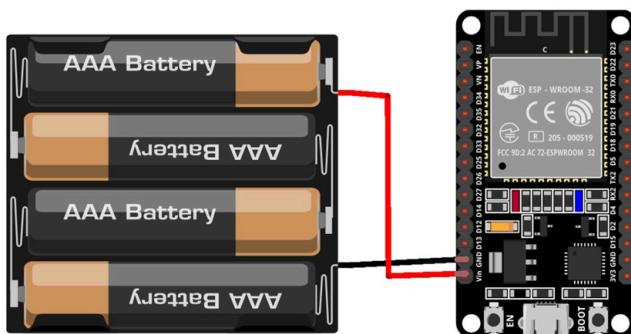
ונלמד כיצד לחבר את הברker לרשת מיד לאחר האתחול.

קישורים:

<https://randomnerdtutorials.com/micropython-wi-fi-manager-esp32-esp8266/>

חשיבות: שימוש במקור מתוך חיצוני

העובדת עם רכיב `h-solo` או רכיב ה-WiFi המובנים בברker ESP32 דורש זרם עזובה גדול מ- 500mA המספק מחיבור ה-USB. על כן יש צורך לחבר מוקור מתוך חיצוני מתאים באוויר



הדק `Vin` ב-ESP32 מחובר למיצב מתח פנימי. על כן ניתן לחבר בהדק `Vin` מתח בין 5V ל-12V. כל מתח בטח זהה בהדק `Vin` עובר למיצב המפחית אותו ל-3.3V ולאחר מכן מזון לצירוד ההיי-קיי של לוח ה-ESP32.

באמצעות סוללה חיצונית של 7V או 9V נוכל להפעיל את ESP32 דרך פין `Vin` על ידי חיבור `GND` של ESP32 עם `GND` של הסוללה. ניתן לחבר כל מתח בין 5V ל-12V לפין `Vin` ESP32 ואולם מומלץ לא להשתמש בו יותר מסוללה חיצונית של 7V. מכיוון ש-ESP32 צריך רק 3.3V כדי לפעול, שאר המתחרים מתפזרים על ידי מיצב המתח כחום.

בדיקות רשתות WiFi זמניות:

להלן דוגמת קוד שבודק מהם רשתות WiFi WiFi הזמינים לשימוש:

```
import network
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)

print("Scanning for WiFi networks, please wait...\n")

authmodes = ['Open', 'WEP', 'WPA-PSK', 'WPA2-PSK4', 'WPA/WPA2-PSK']
for (ssid, bssid, channel, RSSI, authmode, hidden) in sta_if.scan():
    print("* {}".format(ssid))
    print("  - Channel: {}".format(channel))
    print("  - RSSI: {}".format(RSSI))
    print("  - BSSID:")
    {:02x}:{:02x}:{:02x}:{:02x}:{:02x}:{:02x}.format(*bssid)
    print("  - Hidden: {}".format(hidden))
    print()
```

נקבל את הפלט הבא:

```

File Edit Selection View Go Run Terminal Help
Untitled (Workspace)
PYMAKR ...
PROJECTS ...
UART_test ...
Silicon Labs CP210x USB to UART Bridge...
DEVICES ...
Silicon Labs CP210x USB to UART Bridge (...)

main.py 1 x wifimgr.py
11_WiFi_V1 > main.py > ...
1 print("Scanning for WiFi networks, please wait...")
2 print("")
3
4 import network
5 sta_if = network.WLAN(network.STA_IF)
6 sta_if.active(True)
7

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Scanning for WiFi networks, please wait...
* [REDACTED]
- Auth: WPA/WPA2-PSK
- Channel: 11
- RSSI: -30
- BSSID: 54:db:a2:0f:48:81

* [REDACTED]
- Auth: WPA/WPA2-PSK
- Channel: 11
- RSSI: -44
- BSSID: 6e:56:97:d0:b6:8e

* [REDACTED]
- Auth: WPA/WPA2-PSK
- Channel: 1
- RSSI: -65
- BSSID: 18:a6:f7:fe:93:ea

```

חיבור הבקר לרשת ה-Wi-Fi

לאחר שקיבלנו את רשימת הרשאות הזמיןות להתחברות. ניתן לכתוב את הקוד הבא כדי לחבר את הבקר לאחת הרשאות שברשימה:

```

import network

def connect():
    ssid = "שם הרשת"
    password = "סיסמת החיבור לרשת"

    station = network.WLAN(network.STA_IF)

    if station.isconnected() == True:
        print("Already connected")
        print(station.ifconfig())
        return

    station.active(True)
    station.connect(ssid, password)

    while station.isconnected() == False:
        pass

    print("Connection successful")
    print(station.ifconfig())

connect()

```

לאחר הרכבת הקוד נקבל את הקוד הבא:

```

load:0x40078000,len:14888
load:0x40080400,len:3368
entry 0x400805cc
Connection successful
('10.0.0.10', '255.255.255.0', '10.0.0.138', '10.0.0.138')
MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with ESP32
Type "help()" for more information.
>>>
>>>

```

ניתן לראות שהבקר התחבר לשרת וקיבל את כתובת ה- IP המפורטת.

הקבצים `main.py` ו- `boot.py`

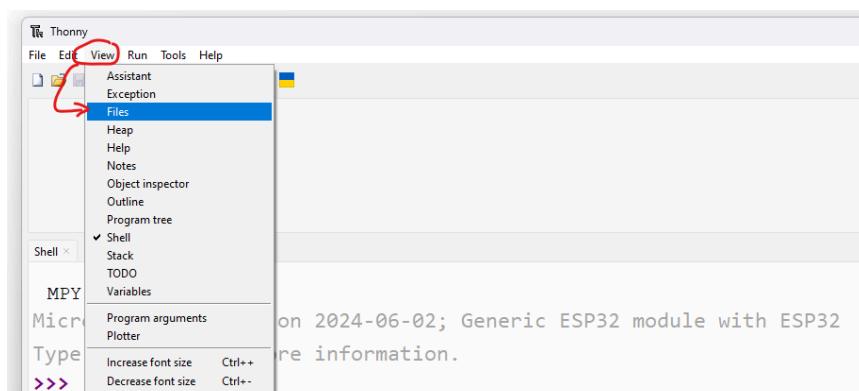
לכל בקר מוגדרים 2 קבצים ייחודיים הנשמרים בו. הראשון קובץ בשם `boot.py` והשני קובץ בשם `main.py`. בקובץ `boot.py` משמש אותנו להקצת קוד עבור הוראות אתחול של הבקר, קוד זה רץ פעמיים אחת בלבד בזמן האתחול. בקובץ זה מתקבל ליבא ספירות רלוונטיות, הגדרת קבועים כמו שמות וסימניות. במשימה זו נעשו שימוש בקובץ זה כדי ליצור לבקר את הקישורית לרשת האינטרנט.

קובץ `main.py` עתיד להכיל את קוד התוכנית שתעבד באופן אוטומטי לאחר הרמת הקובץ `boot.py` קובץ זה ישימוש להפעלת היישום שכתבנו לאחר שלב הפיתוח.

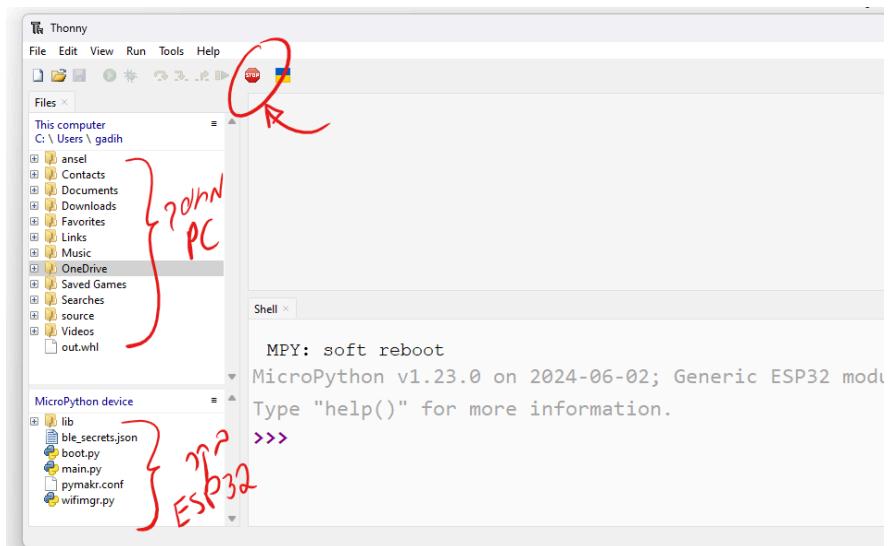
הערה: ברוב הפעמים כאשר כותבים קוד תוכנה עבור בקר בשפת MicroPython תוכלו למצוא שני קבצים המאוחסנים ב.tkern עצמו: הקובץ הראשון נקרא: `boot.py` והשני `main.py`. ברגע שהבקר מתkelig מתח עבודה או מיד לאחר איפוס יוזם, הבקר מפעיל אוטומטית את הקוד השמור בקובץ `boot.py` לאחר מכן הוא יפעיל את תוכן הקובץ `main.py` משמעות הדבר היא שלאחר שמירית הקובץ `main.py` ב.tkern ב.tkern. לא ניתן יהיה לבצע עדכוני תוכנה בו ללא מחיקת הקובץ הנ"ל אליו מדבר באתחול ב.tkern חדש. لكن השימוש באפשרות זו רק כאשר אתם מעבירים את הקוד ממצב פיתוח למצב שימוש.

כתיבת הקובץ `boot.py` לאתחול תקשורת WiFi

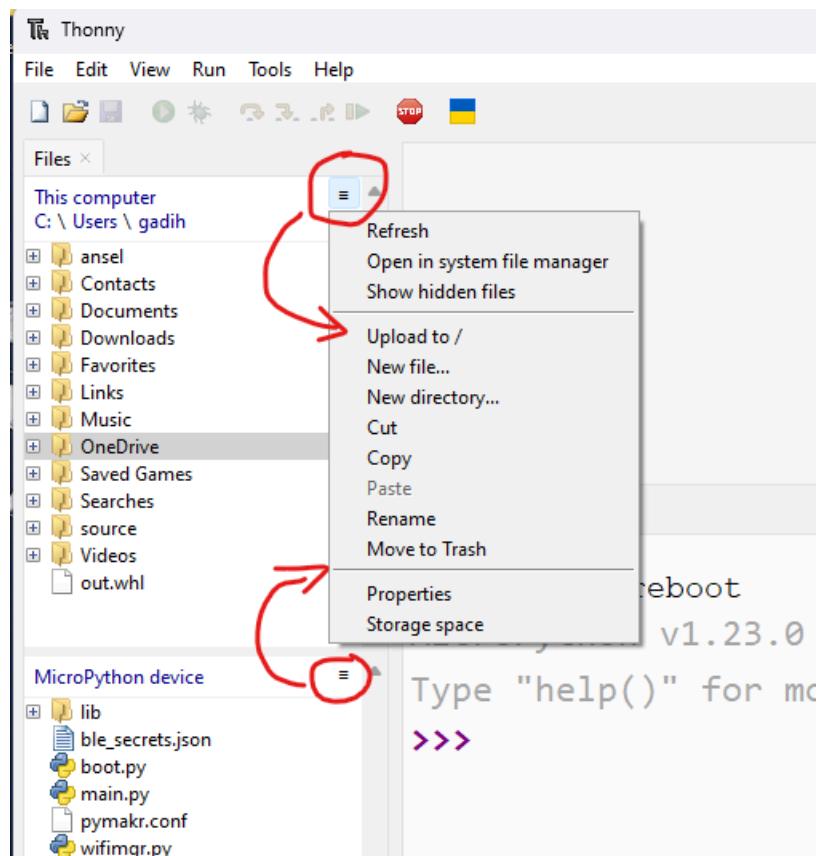
בקר ESP32 מצוייד בזיכרון הרצק בנפח של 4 מגה-בייט שבו ניתן לשמר קבצי קוד ונתונים. נעזר בסביבת הפיתוח Thonny כדי לבדוק את הקבצים שבודיכם הבקר מיד לאחר אתחול ב.tkern בקורסיה חדשה. נעשה זאת על ידי לחיצה על `View` ו- `File` כדי לראות את הקבצים שבבקר:



נפתח לנו המסר הבא:



בעזרת החלון שנפתחה אנו יכולים לראות אילו קבצים שמורים בזיכרון של הבקר. כמו כן ניתן לבצע את כל הפעולות הבסיסיות על הקבצים כמו הוספה קבצים, מחיקת קבצים, יצירה ומחיקה של תיקיות ושינוי של של קובץ.



נעזר במשחק שפתחנו כדי להעתיק את הקובץ `boot.py` מהזיכרון של הבקר למחשב על ידי לחיצה על Upload to

להלן קוד התוכנית **עבור הקיובץ `boot.py`**:

```
# This file is executed on every boot (including wake-boot from deepsleep)
import network

def connect():
    ssid = "yourNetworkName"
    password = "yourNetworkPassword"

    station = network.WLAN(network.STA_IF)

    if station.isconnected() == True:
        print("Already connected")
        print(station.ifconfig())
        return

    station.active(True)
    station.connect(ssid, password)

    while station.isconnected() == False:
        pass

    print("Connection successful")
    print(station.ifconfig())

connect()
```

חשוב: עדכנו בקיובץ את שם רשת ה- Wifi הזמין ואת הסיסמה במקום השורות הבאות:

```
ssid = "yourNetworkName"
password = "yourNetworkPassword"
```

לאחר כתיבת הקובץ `boot.py` נעלם אותו לבקר.

כדי לבדוק שắcן הבקר הת לחבר לאינטרנט נבצע אתחול ונקבל את הפלט הבא:

The screenshot shows the Thonny IDE interface. On the left, there are two file browser panes: 'This computer' and 'MicroPython device'. In the 'MicroPython device' pane, the file 'boot.py' is selected. The main window displays the code in 'boot.py':

```

station = network.WLAN(network.STA_IF)

if station.isconnected() == True:
    print("Already connected")
    return

>>> %Run -c $EDITOR_CONTENT

```

Below the code, the terminal output is shown:

```

MPY: soft reboot
Connection successful
('10.0.0.34', '255.255.255.0', '10.0.0.138', '10.0.0.138')
Already connected
>>>

```

A red circle highlights the text 'Already connected' in the terminal output.

בעקבות חיבורית מוצלחת נקלט בחלון ה- Terminal את מאפייני החיבור מהם:

- כתובת IP של הבקר
- ערך ה- subnet mask
- כתובת ה- gateway
- כתובת ה- DNS

בדיקת כתובת ה- MAC של הבקר

כתובת MAC היא מספר המורכבת מ-48 סיביות שמשמעותו זהות באופן ייחודי רכיב חומרה המחבר לרשת. כתובת MAC מוטבעת בדרך כלל על כרטיס הרשות של המחשב כМОן שבמקרה שלנו כתובת ה- MAC מוטבעת על הבקר עצמו.

כדי לבדוק מה כתובת ה- MAC של הבקר שלנו ניתן להריץ את הקוד הבא:

```

import network
import ubinascii
mac = ubinascii.hexlify(network.WLAN().config('mac'), ':').decode()
print (mac)

```

נקבל פלט הדומה לזה:

```

MPY: soft reboot
Already connected('10.0.0.10', '255.255.255.0', '10.0.0.138', '10.0.0.138')
08:3a:f2:50:ed:d4
>>>

```

כדי לבדוק את התקשרות לאינטרנט ננעזר בקוד הבא:

מקור:

https://docs.micropython.org/en/latest/esp8266/tutorial/network_tcp.html

```

def http_get(url):
    import socket
    _, _, host, path = url.split('/', 3)
    addr = socket.getaddrinfo(host, 80)[0][-1]
    s = socket.socket()
    s.connect(addr)
    s.send(bytes('GET /%s HTTP/1.0\r\nHost: %s\r\n\r\n' % (path, host),
    'utf8'))
    while True:
        data = s.recv(100)
        if data:
            print(str(data, 'utf8'), end='')
        else:
            break
    s.close()

http_get('http://micropython.org/ks/test.html')

```

נקבל את הפלט הבא:

The screenshot shows a development interface with three tabs: 'Files', '[boot.py] x', and '[main.py] * x'. The '[main.py]' tab contains the Python code for the http_get function. A red arrow points from the 'MicroPython device' sidebar to the '[main.py]' tab. The 'MicroPython device' sidebar lists files: lib, ble_secrets.json, boot.py, main.py, pymakr.conf, and wifimgr.py. The 'Shell' tab at the bottom displays the executed code's output, which includes the raw HTTP response headers and the resulting HTML page. A red circle highlights the text 'It's working if you can read this!' within the HTML body.

```

def http_get(url):
    import socket
    _, _, host, path = url.split('/', 3)
    addr = socket.getaddrinfo(host, 80)[0][-1]
    s = socket.socket()
    s.connect(addr)
    s.send(bytes('GET /%s HTTP/1.0\r\nHost: %s\r\n\r\n' % (path, host),
    'utf8'))
    while True:
        data = s.recv(100)
        if data:
            print(str(data, 'utf8'), end='')
        else:
            break
    s.close()

http_get('http://micropython.org/ks/test.html')

Connection: close
Vary: Accept-Encoding
ETag: "529d22da-b4"
Strict-Transport-Security: max-age=15768000
Accept-Ranges: bytes

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Test</title>
  </head>
  <body>
    <h1>Test</h1>
    It's working if you can read this!
  </body>
</html>

```

ניתן לראות שהתחברות מוחזקת לאינטרנט לכתובת <http://micropython.org/ks/test.html> מוחזירה למשוך HTML הכלול את הקוד הבא:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Test</title>
  </head>
  <body>
    <h1>Test</h1>
    It's working if you can read this!
  </body>
</html>
```

משימה 13 - מימוש שרת אינטרנט מבוסס HTTP

בפעולות זו ניצור שרת אינטרנט בסיסי היושב על גבי הבקר. הבקר יקבל גישה לרשת האינטרנט דרך חיבור

WiFi מקומי ויתחיל להאזין לבקשת GET ב프וטוקול HTTP דרך מפתח 80.

להעמקה בנושא פרוטוקול HTTP ניתן לקבל דרך הקישור הבא:

https://he.wikipedia.org/wiki/HTTP_POST

קישורים:

<https://randomnerdtutorials.com/esp32-esp8266-micropython-web-server/>

<https://techtutorialsx.com/2017/06/11/esp32-esp8266-micropython-http-get-requests/>

<https://github.com/micropython/micropython/tree/master/examples/network>

https://github.com/micropython/micropython-esp32/tree/esp32/tests/net_inet

https://github.com/micropython/micropython/blob/master/examples/network/http_server_simplistic.py

לפני ביצוע משימה זו יש לוודא שהබקר מחובר לרשת האינטרנט דרך WiFi כמפורט במשימה 12.

כדי לבדוק שהබкар מחובר לאינטרנט יש לẤתחל את הבקר ולבזק שהשורה הבאה מופיעה:

The screenshot shows a MicroPython REPL window titled 'Shell'. The command '%Run -c \$EDITOR_CONTENT' was run, resulting in the following output:
MPY: soft reboot
Connection successful ✓
('10.0.0.34', '255.255.255.0', '10.0.0.138', '10.0.0.138')
Already connected
The line 'Connection successful' is circled in red, and a red checkmark is placed next to it. The line 'Already connected' is also circled in red.

ישנו מספר גרסאות לישם שרת HTTP בסיסי. נתחיל להדגים זאת על ידי הרשת הći בסיסי שניתן לכתוב בקוד.

חשוב! דוגמה זו מראה כיצד נכתב את שרת-HTTP הקטן ביותר האפשרי ב-MicroPython. אך יש לזכור בחשבון שמדובר בשרת לא מאובטח !!! נdagים זאת:

```
ai = socket.getaddrinfo("0.0.0.0", 8080)
```

משמעות ההוראה היא שהשרת יהיה נגיש לマארחים אחרים ברשת המקומית שלך, ואם לשרת שלך יש חיבור ישיר (ללא חומרת אש) לאינטרנט, אז לכל אחד באינטרנט. עם זאת, היזהרו בעת הפעלת פעללה זו על מחשב המחבר לאינטרנט! החלפו את "0.0.0.0" ב-"127.0.0.1", כדי להפוך את השרת שלכם לנגיש רק למחשב שבו הוא פועל.

במה שקדם הראה כיצד לישם שרת מיועד לשימוש בפרויקטים.

להלן קוד התוכנית עבור שרת HTTP בסיסי (גרסה 1):

```
import socket
```

```

CONTENT = b"""\
HTTP/1.0 200 OK

Hello from MicroPython!
"""

def main():
    s = socket.socket()
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.bind(('0.0.0.0', 8080))
    s.listen(5)
    print("Listening, connect your browser to http://<this_host>:8080/")

    while True:
        res = s.accept()
        client_socket = res[0]
        req = client_socket.recv(4096)
        print("\n\n",req)
        client_socket.send(CONTENT)
        client_socket.close()

main()

```

נקבל את הפלט הבא:

```

>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Already connected('10.0.0.10', '255.255.255.0', '10.0.0.138', '10.0.0.138')
Listening, connect your browser to http://<this_host>:8080/

```

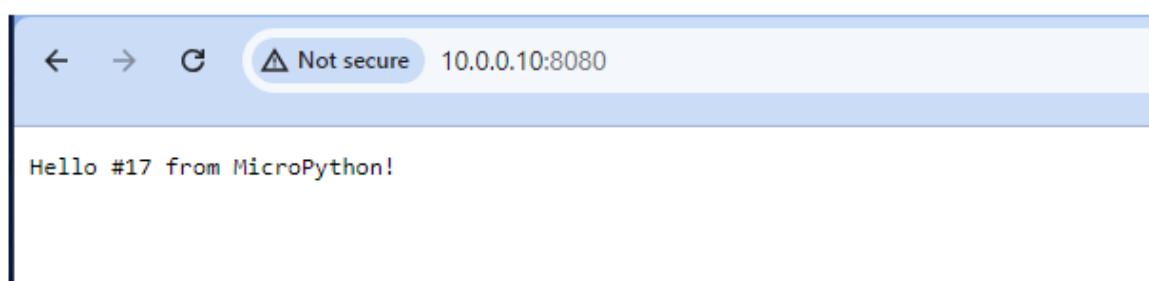
כדי להתחבר לשרת נפתח דפדפן אינטרנט ונכתב את הכתובת

http://<this_host>:8080/

כasher במקום <this_host> נכתבת את כתובות ה- IP כפי שרשומה במסך הפלט:

<http://10.0.0.10:8080/>

נקבל את הפלט הבא:



להלן קוד התוכנית עברו שרת HTTP בסיסי (גרסת 2):

```
import socket
```

```

def html_page():
    html = """
        <!DOCTYPE html>
        <html>
            <head>
                <meta content="width=device-width, initial-scale=1">
            </head>
            <body>
                <h1>Hello world!</h1>
                <p>I am ESP32 in a web server mode</p>
            </body>
        </html>
    """
    return html

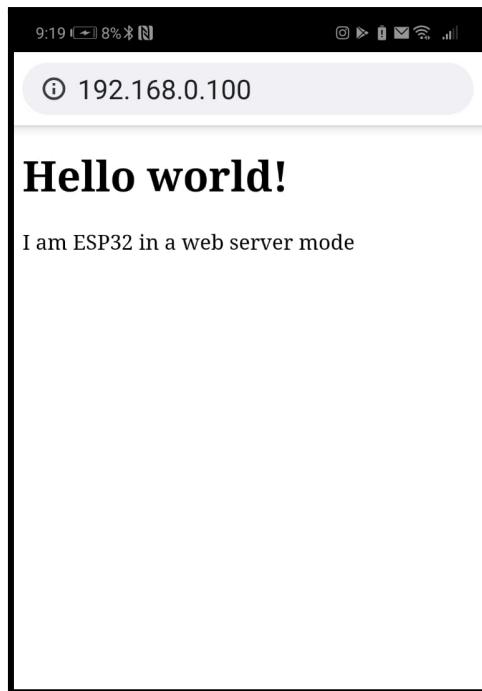
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('0.0.0.0', 8080))
s.listen(5)
print("Listening, connect your browser to http://<this_host>")
while True:
    client_socket, addr = s.accept()
    print("Got a connection from ",addr)
    request = client_socket.recv(1024)
    print("Content = ", request)
    response = html_page()
    client_socket.send("HTTP/1.1 200 OK")
    client_socket.send("Content-Type: text/html;
encoding=utf8\nContent-Length: ")
    client_socket.send(str(len(response)))
    client_socket.send("\nConnection: close\n")
    client_socket.send("\n")
    client_socket.send(response)
    client_socket.close()

```

נגלוש לבקר דרך כתובת ה- IP שלו ונקבל על מסך הדפסן את הפלט הבא:

Hello world!

I am ESP32 in a web server mode



שילוב חומרה בבקר המפעיל שרת WEB - הפעלת נורת LED

נפתח תרגיל המפעיל נורת LED המתחברות להדק 15 של הבקר תוך כדי קליטת הוראות דרך הדפסן.

להלן קוד התוכנית עברו הקובץ תוכנת השרת להפעלת נורת דרך האינטרנט:

```
from machine import Pin
import socket

led = Pin(2, Pin.OUT)

def web_page():
    if led.value() == 1:
        gpio_state="ON"
    else:
        gpio_state="OFF"

    html = """
<html>
    <head>
        <title>ESP32 HTTP Server</title>
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <style>
            html{
                display:inline-block;
                margin: 0px auto;
                text-align:
                center;}
            h1{
                color: #0F3376;
```

```

        padding: 2vh;}
p{
    font-size: 1.5rem;}
button{
    display: inline-block;
    background-color: #3668b8;
    border: none;
    border-radius: 4px;
    color: white;
    padding: 20px 30px;
    font-size: 25px;
}
</style>
</head>
<body>
<h1>ESP32 HTTP Server</h1>
<p>GPIO state: """ + gpio_state + """</p>
<p><a href="/?led=on"><button>LED ON</button></a></p>
<p><a href="/?led=off"><button>LED OFF</button></a></p>
</body>
</html>
"""
return html

s = socket.socket()
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(('0.0.0.0', 8080))
s.listen(5)
print("Listening, connect your browser to http://<this_host>:8080/")

while True:
    client_socket, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = client_socket.recv(1024)
    request = str(request)
    print('Content = %s' % request)

    if '/?led=on' in request:
        print('LED ON')
        led.value(1)
    elif '/?led=off' in request:
        print('LED OFF')
        led.value(0)

    response = web_page()
    client_socket.send('HTTP/1.1 200 OK\n')
    client_socket.send('Content-Type: text/html\n')
    client_socket.send('Connection: close\n\n')
    client_socket.sendall(response)
    client_socket.close()

```

נקבל את הפלט הבא:

```
>>> %Run -c $EDITOR_CONTENT
```

```
MPY: soft reboot
Already connected('10.0.0.10', '255.255.255.0', '10.0.0.138', '10.0.0.138')
Listening, connect your browser to http://<this_host>:8080/
```

כדי להתחבר לשרת נפתח דף אינטרנט ונכתב את הכתובת

http://<this_host>:8080/

כasher במקום <this_host> נכתבת את כתובת ה- IP כפי שרשומה במסך הפלט:

<http://10.0.0.10:8080/>

נקבל את הפלט הבא:

ESP32 HTTP Server

GPIO state: OFF

[LED ON](#)

[LED OFF](#)

שילוב חומרה בברור המפעיל שרת WEB - הצגת מידע מחיישן טמפרטורה LM35

נפתח תרגיל הכלול חיישן טמפרטורה מדגם LM35 המחבר לՀדק 34 של הבקר. בכל פעם שמתחברים לאתר מקבלים קריית נתונים מהחיישן דרך הדף.

להלן קוד התוכנית עברו הקובץ תוכנת השרת להפעלת חיישן טמפרטורה דרך האינטרנט:

```
import socket
from machine import ADC, Pin

SENSOR_ADC_PIN = 34
sensor_pin = Pin(SENSOR_ADC_PIN)
adc = ADC(sensor_pin)
adc.atten(ADC.ATTN_11DB)

def read_temperature():
    raw_value = adc.read()
    voltage = (raw_value / 4095) * 3.3
    temperature_celsius = voltage * 100
    return temperature_celsius
```

```

def start_temperature_server():
    s = socket.socket()
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.bind(('0.0.0.0', 8080))
    s.listen(5)
    print("Listening, connect your browser to http://<this_host>:8080/")

    while True:
        client_socket, client_address = s.accept()
        print("Client address:", client_address)
        try:
            request = client_socket.recv(1024)
            print("Request:")
            print(request)

            current_temperature = read_temperature()

            html_page_temp = """<!DOCTYPE html>
<html>
<head>
    <title>Temperature sensor</title>
    <style>
        body {{
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            min-height: 100vh;
            margin: 0;
            font-family: Arial, sans-serif;
        }}
        .container {{
            padding: 20px;
            text-align: center;
            border: 1px solid #ccc;
            border-radius: 5px;
        }}
        h1 {{
            font-size: 1.8em;
            margin-bottom: 0.8em;
            color: #333;
        }}
        p {{
            font-size: 1.1em;
            margin-bottom: 0.5em;
            color: #555;
        }}
        strong {{
            color: #000;
        }}
    </style>
</head>
<body>
<div class="container">

```

```

<h1>Temperature sensor</h1>
<p>Current temperature: <strong>{:.2f} &deg;C</strong></p>
</div>
</body>
</html>
""".format(current_temperature)

response = "HTTP/1.0 200 OK\r\nContent-Type: text/html\r\n\r\n"
+ html_page_temp
    client_socket.sendall(response)

except OSError as e:
    print("OSError:", e)
finally:
    client_socket.close()
print("Client socket close.\n")

start_temperature_server()

```

נקבל את הפלט הבא:

```

>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Already connected
('10.0.0.24', '255.255.255.0', '10.0.0.138', '10.0.0.138')
Listening, connect your browser to http://<this_host>:8080/

```

כדי להתחבר לשרת נפתח דף אינטרנט ונכתב את הכתובת

http://<this_host>:8080/

כasher במקום <this_host> נכתבת את כתובות ה- IP כפי שרשומה במסך הפלט:

<http://10.0.0.24:8080/>

נקבל את הפלט הבא:

Temperature sensor

Current temperature: **27.24 °C**

משימה 14 - HTTP GET

בפעילות זו נלמד כיצד להשתמש בבקר קלוקו WiFi. הבקר יוכל גישה לרשת האינטרנט דרך חיבור WiFi מקומי ובכך הוא יוכל לקבל ולשלוח נתונים לאינטרנט ממש כמו שדרפן במחשב או טלפון נייד לעשוות. כל התקשרות תעבור תחת בקשות GET בפרוטוקול HTTP דרך מפתח 80.

להעמקה בנושא פרוטוקול HTTP ניתן לקבל דרך הקישור הבא:

https://he.wikipedia.org/wiki/HTTP_POST

קישורים:

<https://randomnerdtutorials.com/esp32-esp8266-micropython-web-server/>

<https://techtutorialsx.com/2017/06/11/esp32-esp8266-micropython-http-get-requests/>

<https://github.com/micropython/micropython/tree/master/examples/network>

https://github.com/micropython/micropython-esp32/tree/esp32/tests/net_inet

לפני ביצוע משימה זו יש לוודא שהבקר מחובר לרשת האינטרנט דרך WiFi כמפורט במשימה 12.

כדי לבדוק שהבקר מחובר לאינטרנט יש לאותחל את הבקר ולבדק שהשורה הבאה מופיע:

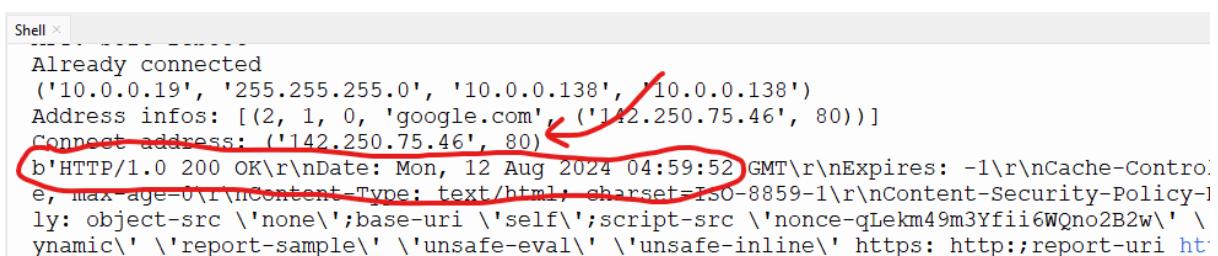


```
Shell >>> %Run -c $EDITOR_CONTENT
>>> MPY: soft reboot
Connection successful ✓
('10.0.0.34', '255.255.255.0', '10.0.0.138', '10.0.0.138')
Already connected
>>>
```

להלן קוד התוכנית התחבר לאתר של גוגל ומציג את המידע המתקבל כמחרוזת טקסט על מסך הטרמינל:

```
import socket
s = socket.socket()
ai = socket.getaddrinfo("google.com", 80)
print("Address infos:", ai)
addr = ai[0][-1]
print("Connect address:", addr)
s.connect(addr)
s.send(b"GET / HTTP/1.0\r\n\r\n")
print(s.recv(4096))
s.close()
```

פלט התוכנית יהיה כך:



```
Shell >>> .....
Already connected
('10.0.0.19', '255.255.255.0', '10.0.0.138', '10.0.0.138')
Address infos: [(2, 1, 0, 'google.com', ('142.250.75.46', 80))]
Connect address: ('142.250.75.46', 80) ↗
b'HTTP/1.0 200 OK\r\nDate: Mon, 12 Aug 2024 04:59:52 GMT\r\nExpires: -1\r\nCache-Control: max-age=0\r\nContent-Type: text/html; charset=ISO-8859-1\r\nContent-Security-Policy: object-src \'none\';base-uri \'self\';script-src \'nonce-qLekm49m3Yfii6WQno2B2w\' \\'dynamic\' \\'report-sample\' \\'unsafe-eval\' \\'unsafe-inline\' https: http://report-uri ht...
```

מצד אחד ניתן לראות שבקר ESP32 מצליח להתחבר לכל אתר אינטרנט מצד שני לפי מבנה התשובה לא נראה שנית להשתמש בתנונים שהתקבלו באופן שימושי.

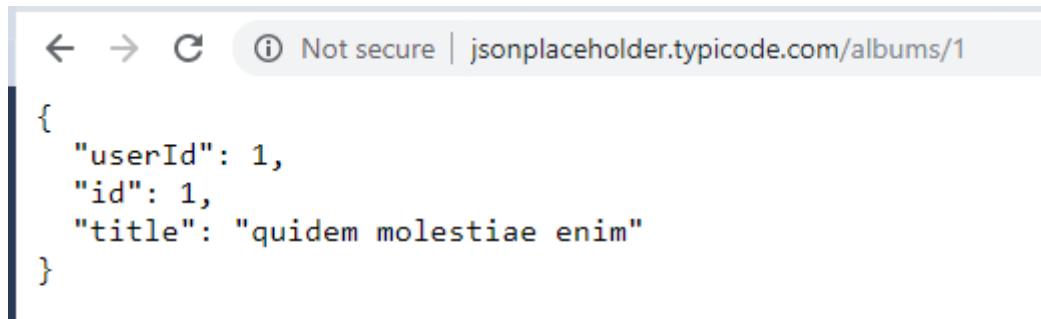
במטרה לנצל את יכולת הקישוריות לאינטרנט בצורה טוביה יותר נעשה שימוש בקבלת מבנה נתונים מסווג JSON הנראית כ:

{name:"Gadi", age:50, city:"Migdal HaEmek"}

ගלשו לכתובת האינטרנט הבא:

<http://jsonplaceholder.typicode.com/albums/1>

תקבלו את המסר הבא:



The screenshot shows a browser window with the URL "jsonplaceholder.typicode.com/albums/1". The page displays a single JSON object:

```
{  
  "userId": 1,  
  "id": 1,  
  "title": "quidem molestiae enim"  
}
```

בקר ESP32 מסוגל להתחבר לאתר כדי לקבל את מבנה הנתונים בפורמט JSON ולפרק אותו.

שילוב JSON עם MicroPython

נדגים קוד העושה שימוש במחלקה JSON כדי לקרוא קובץ טקסט בשם `json.data` הכיל את התוכן הבא:

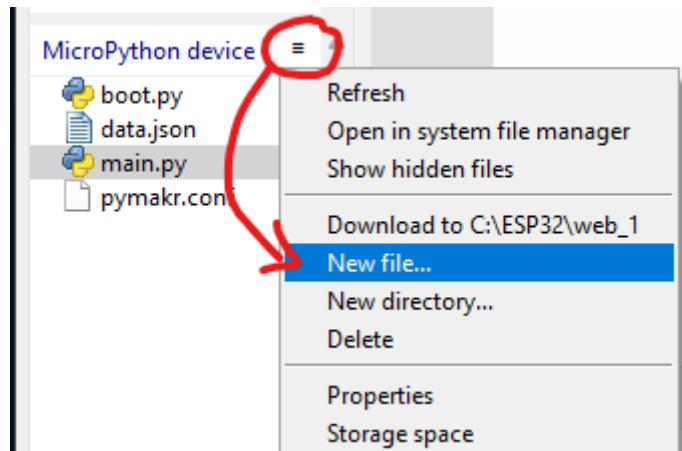
```
{"is_led_on": 1}
```

דוגמת הקוד מראה כיצד לקרוא את הקובץ. להמיר את התוכן שלו למבנה נתונים ולהשתמש מיד עם LED בהתאם לערך השמור בקובץ.

```
import json  
from machine import Pin  
  
# load the config file from flash  
f=open("data.json")  

```

נדגש שיש ליצור קובץ בשם `json.data` השמור בתוך הAKER. כדי לעשות זאת יש ללחוץ כמפורט בתמונה:



נשלב עכשו את הקוד המאפשר להתחבר לאתר ייחד עם הקוד שפענה JSON כדי להתחבר לאטר אינטרנט הכלול מידע בפורמט JSON לקרוא אותו ולהשתמש במידע שרשום בו. אך הפעם ניעיל את הקוד ונעשה שימוש במחלקה `urequests` המאפשרת גם להתחבר לאתר וגם לפענה את המידע בפורמט JSON. להלן הקוד:

```
import urequests
response = urequests.get('http://jsonplaceholder.typicode.com/albums/1')
print(type(response))
print(response.text)
print(type(response.text))
parsed = response.json()
print(type(parsed))
print(parsed["userId"])
print(parsed["id"])
print(parsed["title"])
```

ניתן לראות שהבוקר קלט את הנתונים והצליח לבדוק כל אחד מהם להמשך טיפול

```
MPY: soft reboot
Already connected
('10.0.0.19', '255.255.255.0', '10.0.0.138', '10.0.0.138')
<class 'Response'>
{
    "userId": 1,
    "id": 1,
    "title": "quidem molestiae enim"
}
<class 'str'>
<class 'dict'>
1
1
quidem molestiae enim
```

ישום שירות API לצורך לקבלת נתונים מזג אוויר

דמיינו שאתה במסעדה ואתם רוצח להזמין אוכל. במקומות למכת למטבח ולנסות להcin את האוכל בעצמך, אתה פשוט קוראים למלצר ומקשימים ממנו את מה שאתה רוצה. המלצר הוא כמו ה-`API`. הוא מקבל את הזמן מה לקוחות (הבקשה), מעביר אותה למטבח (השירות) ומחזיר לכם את האוכל (התשובה).

از מה זה API בעצם?

API זה כמו שליח שמאפשר לישומים שונים לתקשר ביניהם ולהחליף מידע. במקומות שבהם ישום יבנה את כל התכונות שלו מ一封, הוא יכול להשתמש ב-API של ישום אחר כדי לקבל גישה לתוכנות האלה.

דוגמאות:

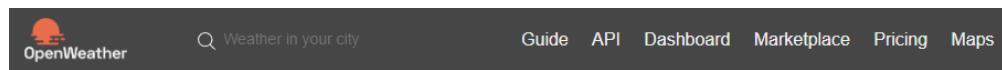
مפות גוגל: כשאתם משתמשים באפליקציה של מונית, האפליקציה משתמשת ב-API של מפות גוגל כדי להציג לכם את המיקום שלכם ואת היעד.

רשויות חברותיות: כשאתם נכנסים לאתר באמצעות חשבון פייסבוק, האתר משתמש ב-API של פייסבוק כדי לאמת את הזהות שלכם.

נדגים שירות API של <https://openweathermap.org> כדי לקבל נתונים מזג אוויר עכשוויים במקום מסוים.

כדי לקבל גישה לשירותי ה-API אנו זקוקים להירשם באתר ולקבל קוד ID ייחודי שב Zukunft נקבל גישה לנתונים מזג האוויר. שירות זה בחינם לשימושים בסיסיים כמו>Status. נתחבר לאתר הבא:

<https://openweathermap.org/appid/>



Best way to start and continue calling OpenWeather APIs

OpenWeather platform is a set of elegant and widely recognisable APIs. Powered by convolutional machine learning solutions, it is capable of delivering all the weather information necessary for decision making for any location on the globe. To start using our APIs, please sign up [here](#).

Why our Free Weather API is so good yet free

How to call OpenWeather APIs with a freemium plan

The API key is all you need to call any of our weather APIs. Once you [sign up](#) using your email, the API key (APPID) will be sent to you in a confirmation email. Your API keys can always be found on your [account page](#), where you can also generate additional API keys if needed. Check our [documentation page](#) to find all technical information for each product. Documentation is an essential guide with actual examples and comprehensive description of API calls, responses and parameters.

לאחר ההרשמה נקבל קוד ID כמתואר כאן:

לאחר קבלת הקוד אנו יכולים לשלב אותו בכתובת אינטרנט URL ולקבל JSON הכלל את נתוני מזג האוויר, באופן הבא:

<https://api.openweathermap.org/data/2.5/weather?lat=32.811&lon=35.012&appid=99e5XXXXX6142>

כasher הערכים lat ו- lon מצינים את קו האורך והרוחב של הנקודה שבה אתם רוצים לקבל את נתוני מזג האוויר. ו- appid הוא הקוד ה- ID שלכם.

נדגים זאת על ידי גישה ישירה לכתובת תוך שימוש בדף:

נשלב עכשו את כל מה שלמדנו בפרק זה כדי לבדוק את נתוני מזג האוויר מתוך ה- JSON שקיבלנו:

```
import urequests
response =
urequests.get('https://api.openweathermap.org/data/2.5/weather?lat=32.000&lon=35.000&appid=99eXXXXXX6142')
#print(type(response))
#print(response.text)
#print(type(response.text))
parsed = response.json()
#print(type(parsed))
#print(parsed["main"])
#print(parsed["main"]["temp_max"] - 273.15)
print("temperature: ", parsed["main"]["temp"] - 273.15)
#print(parsed["main"]["temp_min"] - 273.15)
print("humidity: ", parsed["main"]["humidity"])
```

נקבל את הפלט הבא:

משימה 15 - הפעלת צג גרפי דגם SSD1306 OLED display

במשימה זה תלמד כיצד להשתמש בתצוגת OLED SSD1306 בגודל 0.96 אינץ' המתחברת לבקר ESP32. במהלך המשימה נלמד להציג הודעות טקסט למרוחם שמדובר במסך גרפי.

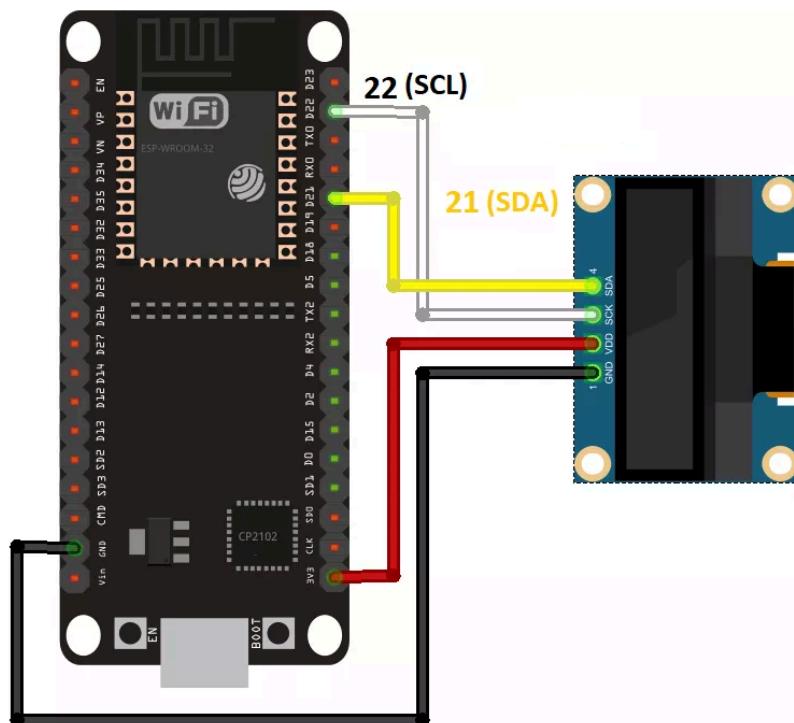
קישורים:

<https://randomnerdtutorials.com/micropython-oled-display-esp32-esp8266/>

חיבור הצג לבקר

OLED	ESP32
Vin	3.3V
GND	GND
SCL	GPIO 22
SDA	GPIO 21

شرطוט חשמלי של החיבור:



```

import time
import framebuf

# register definitions
SET_CONTRAST      = const(0x81)
SET_ENTIRE_ON     = const(0xa4)
SET_NORM_INV      = const(0xa6)
SET_DISP          = const(0xae)
SET_MEM_ADDR      = const(0x20)
SET_COL_ADDR      = const(0x21)
SET_PAGE_ADDR     = const(0x22)
SET_DISP_START_LINE = const(0x40)
SET_SEG_REMAP      = const(0xa0)
SET_MUX_RATIO      = const(0xa8)
SET_COM_OUT_DIR    = const(0xc0)
SET_DISP_OFFSET    = const(0xd3)
SET_COM_PIN_CFG    = const(0xda)
SET_DISP_CLK_DIV    = const(0xd5)
SET_PRECHARGE      = const(0xd9)
SET_VCOM_DESEL     = const(0xdb)
SET_CHARGE_PUMP    = const(0x8d)

class SSD1306:
    def __init__(self, width, height, external_vcc):
        self.width = width
        self.height = height
        self.external_vcc = external_vcc
        self.pages = self.height // 8
        # Note the subclass must initialize self.framebuf to a framebuffer.
        # This is necessary because the underlying data buffer is different
        # between I2C and SPI implementations (I2C needs an extra byte).
        self.poweron()
        self.init_display()

    def init_display(self):
        for cmd in (
            SET_DISP | 0x00, # off
            # address setting
            SET_MEM_ADDR, 0x00, # horizontal
            # resolution and layout
            SET_DISP_START_LINE | 0x00,
            SET_SEG_REMAP | 0x01, # column addr 127 mapped to SEG0
            SET_MUX_RATIO, self.height - 1,
            SET_COM_OUT_DIR | 0x08, # scan from COM[N] to COM0
            SET_DISP_OFFSET, 0x00,
            SET_COM_PIN_CFG, 0x02 if self.height == 32 else 0x12,
            # timing and driving scheme
            SET_DISP_CLK_DIV, 0x80,
            SET_PRECHARGE, 0x22 if self.external_vcc else 0xf1,
            SET_VCOM_DESEL, 0x30, # 0.83*Vcc
            # display
            SET_CONTRAST, 0xff, # maximum
            SET_ENTIRE_ON, # output follows RAM contents
            SET_NORM_INV, # not inverted
            # charge pump

```

```

        SET_CHARGE_PUMP, 0x10 if self.external_vcc else 0x14,
        SET_DISP | 0x01): # on
        self.write_cmd(cmd)
    self.fill(0)
    self.show()

def poweroff(self):
    self.write_cmd(SET_DISP | 0x00)

def contrast(self, contrast):
    self.write_cmd(SET_CONTRAST)
    self.write_cmd(contrast)

def invert(self, invert):
    self.write_cmd(SET_NORM_INV | (invert & 1))

def show(self):
    x0 = 0
    x1 = self.width - 1
    if self.width == 64:
        # displays with width of 64 pixels are shifted by 32
        x0 += 32
        x1 += 32
    self.write_cmd(SET_COL_ADDR)
    self.write_cmd(x0)
    self.write_cmd(x1)
    self.write_cmd(SET_PAGE_ADDR)
    self.write_cmd(0)
    self.write_cmd(self.pages - 1)
    self.write_framebuf()

def fill(self, col):
    self.framebuf.fill(col)

def pixel(self, x, y, col):
    self.framebuf.pixel(x, y, col)

def scroll(self, dx, dy):
    self.framebuf.scroll(dx, dy)

def text(self, string, x, y, col=1):
    self.framebuf.text(string, x, y, col)

class SSD1306_I2C(SSD1306):
    def __init__(self, width, height, i2c, addr=0x3c, external_vcc=False):
        self.i2c = i2c
        self.addr = addr
        self.temp = bytearray(2)
        # Add an extra byte to the data buffer to hold an I2C data/command byte
        # to use hardware-compatible I2C transactions. A memoryview of the
        # buffer is used to mask this byte from the framebuffer operations
        # (without a major memory hit as memoryview doesn't copy to a separate
        # buffer).
        self.buffer = bytearray(((height // 8) * width) + 1)
        self.buffer[0] = 0x40 # Set first byte of data buffer to Co=0, D/C=1
        self.framebuf = framebuffer.FrameBuffer1(memoryview(self.buffer)[1:], width,
height)
        super().__init__(width, height, external_vcc)

```

```

def write_cmd(self, cmd):
    self.temp[0] = 0x80 # Co=1, D/C#=0
    self.temp[1] = cmd
    self.i2c.writeto(self.addr, self.temp)

def write_framebuf(self):
    # Blast out the frame buffer using a single I2C transaction to support
    # hardware I2C interfaces.
    self.i2c.writeto(self.addr, self.buffer)

def poweron(self):
    pass

class SSD1306_SPI(SSD1306):
    def __init__(self, width, height, spi, dc, res, cs, external_vcc=False):
        self.rate = 10 * 1024 * 1024
        dc.init(dc.OUT, value=0)
        res.init(res.OUT, value=0)
        cs.init(cs.OUT, value=1)
        self.spi = spi
        self.dc = dc
        self.res = res
        self.cs = cs
        self.buffer = bytearray((height // 8) * width)
        self.framebuf = framebuffer.FrameBuffer1(self.buffer, width, height)
        super().__init__(width, height, external_vcc)

    def write_cmd(self, cmd):
        self.spi.init(baudrate=self.rate, polarity=0, phase=0)
        self.cs.high()
        self.dc.low()
        self.cs.low()
        self.spi.write(bytearray([cmd]))
        self.cs.high()

    def write_framebuf(self):
        self.spi.init(baudrate=self.rate, polarity=0, phase=0)
        self.cs.high()
        self.dc.high()
        self.cs.low()
        self.spi.write(self.buffer)
        self.cs.high()

    def poweron(self):
        self.res.high()
        time.sleep_ms(1)
        self.res.low()
        time.sleep_ms(10)
        self.res.high()

```

תוכן הקובץ `yd.run`:

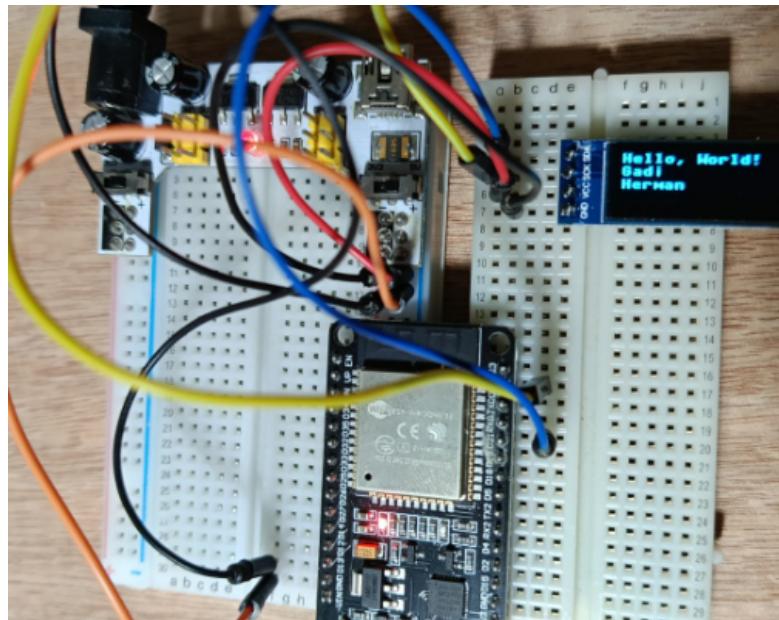
```
from machine import Pin, I2C
import ssd1306
from time import sleep

# using default address 0x3C
i2c = I2C(scl=Pin(22), sda=Pin(21))
oled_width = 128
oled_height = 32
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)

oled.text('Hello, World!', 0, 0)
oled.text('Gadi', 0, 10)
oled.text('Herman', 0, 20)

oled.show()
```

להלן הפלט על הצג:



משימה 16 - שירותי ענן מבוססי MQTT

קישורים:

<https://github.com/miketeachman/micropython-adafruit-mqtt-esp8266>

<https://io.adafruit.com/api/docs/mqtt.html#adafruit-io-mqtt-api>

<https://www.emqx.com/en/blog/micro-python-mqtt-tutorial-based-on-raspberry-pi>

<https://www.hackster.io/mark-yu/air-quality-system-with-beebotte-and-ifttt-2922c7>

MQTT - Message Queuing Telemetry Transport הוא פרוטוקול תקשורת קל משקל המשמש בעיקר לתקשורת בין מכשירים באינטרנט (TCP). להלן תיאור העקרונות שלו:

1. MQTT פועל על בסיס מודל "פרסום-הרשם" (publish-subscribe). במודל זה, מכשירים יכולים לשלוח ("פרסם") הודעות לנושאים (Topics), ומיכשרים אחרים יכולים "להירשם" (subscribe) לנושאים (Topics) שמשמעותם אותם כדי לקבל את ההודעות הללו.

2. רכיבים עיקריים:

לקוחות: אלו הם המכשירים או התוכנות שמספרמים הודעות או נרשמיים לקבלתן.

ברוקר (מתווך): זהו השירות המרכזי שמנהל את כל התקשורת בין הלקוחות.

3. נושאים (Topics):

כל הודעה ב-MQTT משיכת לנושא מסוים.

נושאים (Topics) מאורגנים בצורה היררכית, כמו מבנה תיקיות.

לדוגמה: "בית/סלון/טפרטורה" יכול להיות נושא למדידת טפרטורה בסלון.

4. יתרונות:

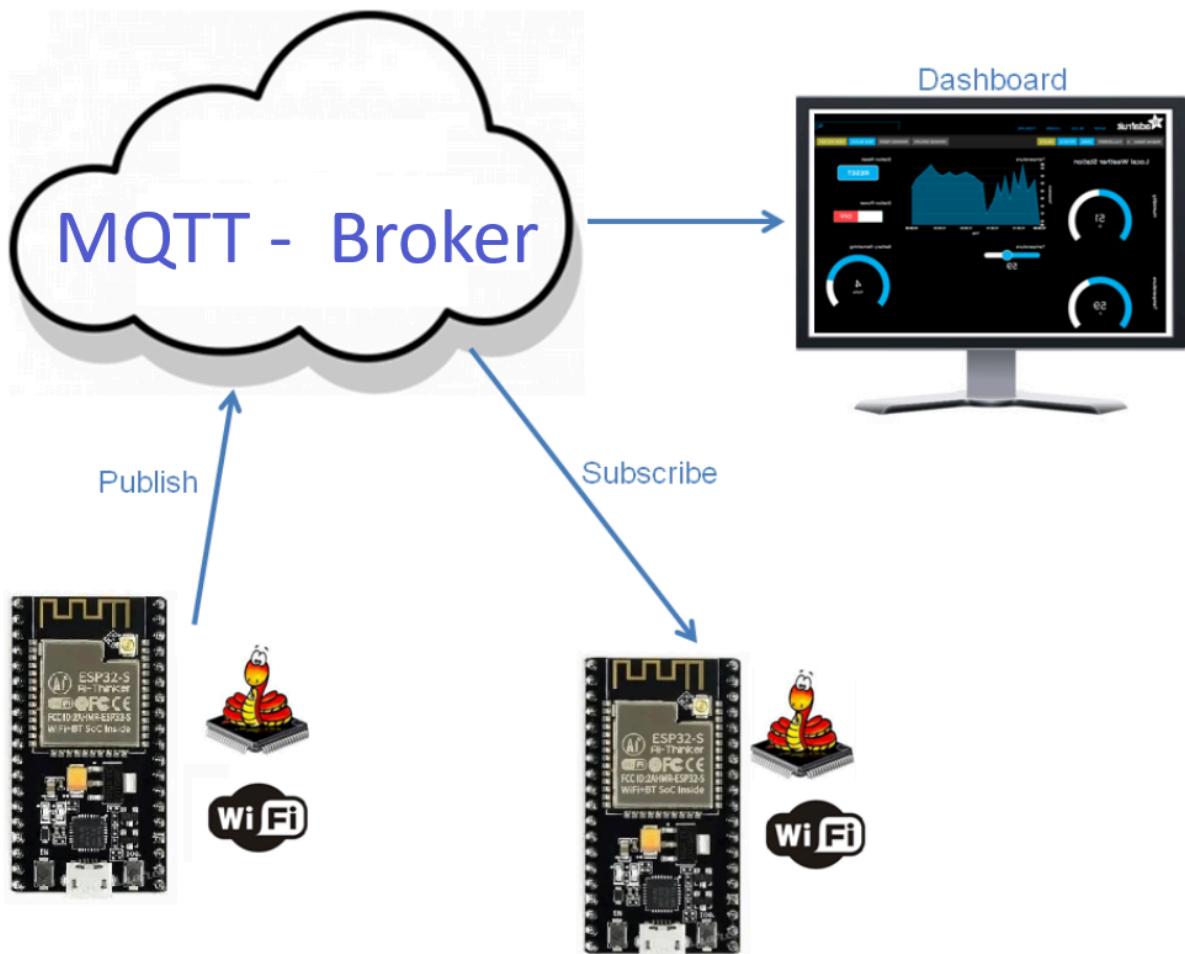
הפרוטוקול צריך מעט משאבי רשות וחומרה, מה שמתאים מאד למיכשי IoT קטנים.

מתאים למיכשרים המופעלים על סוללה.

מאפשר תקשורת כמעט בזמן אמת.

כולל מנגנונים להבטחת העברת הודעות גם ברשות לא יציבות.

5. שימושים נפוצים: מערכות בית חכם, ניטור מכונות בתעשייה, מעקב אחר צי רכבים, איסוף נתונים מחישנים מרוחקים.



יצירת חשבון באתר beebotte.com

באתר beebotte הוא שירות ענן המיועד לפROYקטים של אינטרנט הדברים (IoT). להלן סקירה כללית על השירות שמציע האתר:

- מאפשרת לחבר מכשיר IoT לענן וניהל אותו מרוחק.
- מאפשר ליצור ממשקי משתמש מותאמים אישית לשיליטה וניהול של מכשיר IoT. כולל אלמנטים כמו כפתורים, מד מתח וגרפים.
- מאפשר לאחסן וניהל נתונים מחישנים או מכשירים.
- מספק ברוקר MQTT לתקשורת ייעילה בין מכשירים.
- מאפשר להציג פעולות אוטומטיות בהתאם על תנאים מסוימים. לדוגמה, שליחת התראה כאשר טמפרטורה עולה מעל סף מסוים.
- מציע תוכנית חינמית עם מגבלות מסוימות, ותוכניות בתשלום עם יכולות נרחבות יותר.

נתחבר לאתר הבא ונפתח בו חשבון משתמש:

<https://beebotte.com/>

לאחר התחברות מוצלחת נקבל את המסר הבא:

The screenshot shows the 'My Channels' section of the beebootte.com web interface. On the left is a sidebar with navigation links: Channels, Dashboards, Beerules (beta), Console, Account Settings, Account Usage, and Support. The main area displays four channels:

- esp32**: Created by gadiherman, Private, Created: August 24th 2024.
- Multipass1**: Created by gadiherman, Private, Created: August 23rd 2024.
- temp1**: Created by gadiherman, Private, Created: August 23rd 2024.
- Multipass**: Created by gadiherman, Private, Created: August 23rd 2024.

שימוש באתר beebootte.com במטרה לשלוט על רכיב חומרה

בחלק זה של הפעולות נדגים כיצד יוצריםلوح בקירה הכלול לחץ שמליך ומכבה LED בחומרה.

নিচৰ তচিলা ছাই নিচৰ তচিলা ছাই

The screenshot shows the 'Create a new channel' form. The sidebar on the left is identical to the previous one. The main form has the following fields:

- Name:** els32 (highlighted with a red circle)
- Description:** Channel Description (empty)
- Visibility:** Public
- Configured Resources:**
 - Resource Name: led (highlighted with a red circle)
 - Resource Type: boolean (highlighted with a red circle)
 - SoS: SoS (highlighted with a red circle)
- Buttons:** Cancel (gray) and Create channel (green, highlighted with a red circle)

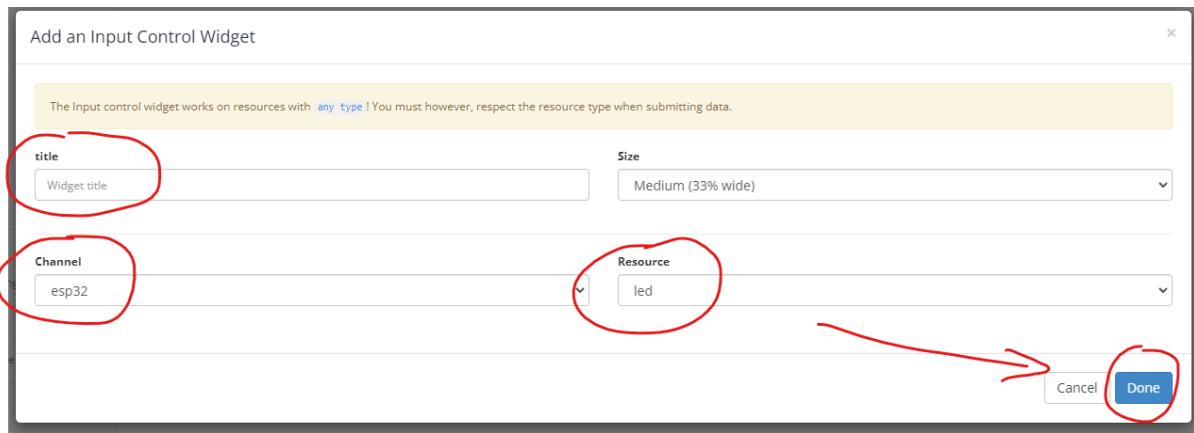
בשלב הבא ניצורلوح בקירה חדש בשם LedControl

TITLE	DESCRIPTION	CREATED ON	SCOPE	VIEWS
My Dashboard		August 21st 2024	Public	64

נכנו לעיצוב הממשק על ידי לחיצה של Add Widget

נוסף בו לחץן על ידי לחיצה על Create New Block

בשלב הבא נקשר בין הלחצן שייצרנו עם ה- Channel שמו led/esp32 שיצרנו מקודם:



נקבל את הlion הבא:

הדבר האחרון שהוא נדרש הוא כדי לחבר בין האתר שמספק לנו שירות ענן לבין הילד שמחובר לבקר ESP32 הוא קוד הגישה הייחודי ל- Channel. נקבל אותו על ידי ה- Channel Token.

נפתח את סביבת התכנות כדי לכתוב את הקוד הבא:

הקובץ boot.py

```
# This file is executed on every boot (including wake-boot from deepsleep)
import network

def connect():
    ssid = "XXXX"
    password = "XXXX"
    station = network.WLAN(network.STA_IF)
    if station.isconnected() == True:
        print("Already connected")
        return
```

```

station.active(True)
station.connect(ssid, password)
while station.isconnected() == False:
    pass
print("Connection successful")
print(station.ifconfig())
connect()

```

להלן הגרסת ה-CI פשוטה שניתן לכתוב ב.tk (גרסת למודית).

הקובץ main.py

```

from umqtt.simple import MQTTClient
import ujson
import utime

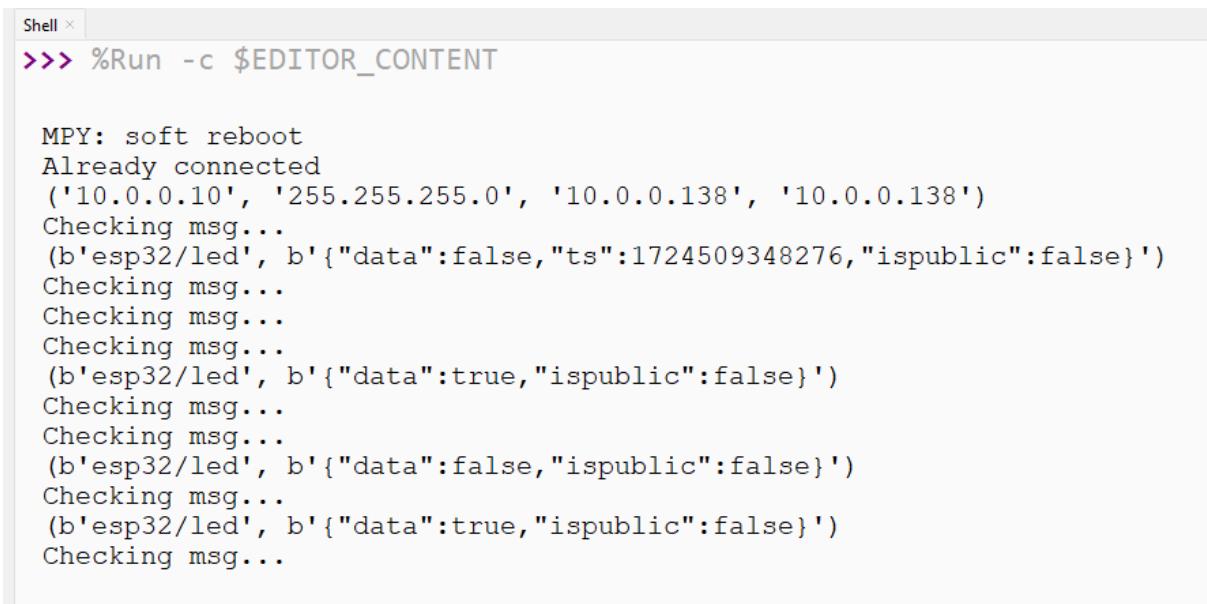
def callback_func(topic, msg):
    print((topic, msg))

client = MQTTClient("test_mqtt_client_id", 'mqtt.beebotte.com',
user='token:token_jraXXXXXXXXXXXXku', password='', keepalive=30)
client.connect()
client.set_callback(callback_func)
client.subscribe('esp32/led')

while True:
    print("Checking msg...")
    client.check_msg()
    utime.sleep(1)

```

לאחר צריבת הרכיב וחיבורו לאינטרנט על ידי תקשורת WiFi בכל פעם שנלחץ על הלוחץ שבאτר נקלט ב.tk
את ההודעה הבא:



```

Shell < 
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Already connected
('10.0.0.10', '255.255.255.0', '10.0.0.138', '10.0.0.138')
Checking msg...
(b'esp32/led', b'{"data":false,"ts":1724509348276,"ispublic":false}')
Checking msg...

```

כל מה שנשאר לעשות זה להוסיף לפועלה callback_func את הקוד הבא כדי להפעיל בפועל את ה- LED בהתאם לפוקודה שקיבל. נדגים זאת:

```
from machine import Pin
from umqtt.simple import MQTTClient
import ujson
import utime

LED = Pin(2, Pin.OUT)

def callback_func(topic, msg):
    print((topic, msg))
    json_data= ujson.loads(msg)
    dt= json_data["data"]
    if str(dt) == 'True':
        LED.value(1)
    if str(dt) == 'False':
        LED.value(0)

client = MQTTClient("test_mqtt_client_id", 'mqtt.beebotte.com',
user='token:token_jraXXXXXXXXXUku', password='', keepalive=30)
client.connect()
client.set_callback(callback_func)
client.subscribe('esp32/led')

while True:
    print("Checking msg...")
    client.check_msg()
    utime.sleep(1)
```

להלן גרסה יציבה יותר לתקשורת. גרסה המיעדרת לשילוב בפרויקטים:

```
from machine import Pin, Timer
from umqtt.simple import MQTTClient
import ujson
import sys
import os

LED = Pin(2, Pin.OUT)
PING_PERIOD = 120

CHANNEL_TOKEN = 'token_jraXXXXXXXXXUku'
CHANNEL_NAME = 'esp32'
RESOURCE_NAME = 'led'
MQTT_SERVER = 'mqtt.beebotte.com'
MQTT_USER = 'token:' + CHANNEL_TOKEN
MQTT_TOPIC = CHANNEL_NAME + '/' + RESOURCE_NAME

def handleTimerInt(timer):
    client.ping()
    print('ping')
```

```

def callback_func(topic, msg):
    print("topic:",topic," msg:", msg)
    json_data= ujson.loads(msg)
    dt= json_data["data"]
    print("*** " + str(dt) + " ***")
    if dt:
        LED.value(1)
    else:
        LED.value(0)

# create a random MQTT clientID
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_'+str(random_num), 'utf-8')

client = MQTTClient(mqtt_client_id, MQTT_SERVER, user=MQTT_USER,
password='', keepalive=PING_PERIOD*2 )

myTimer = Timer(0)

try:
    client.connect()
    myTimer.init(period=PING_PERIOD*1000, mode=Timer.PERIODIC,
callback=handleTimerInt)
except Exception as e:
    print('could not connect to MQTT server {}'.format(type(e).__name__, e))
    sys.exit()

client.set_callback(callback_func)
client.subscribe(MQTT_TOPIC)

while True:
    try:
        client.wait_msg()
    except KeyboardInterrupt:
        print('Ctrl-C pressed...exiting')
        client.disconnect()
        sys.exit()

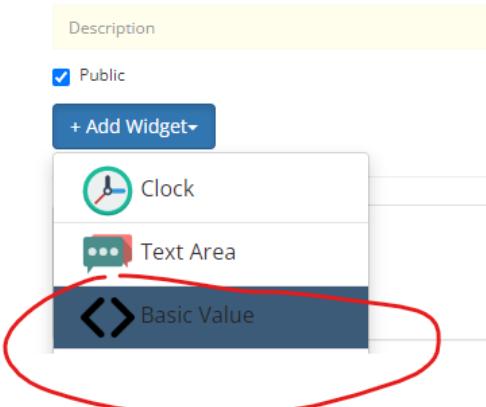
```

שימוש באתר beebotte.com במטרה לקלוט מידע מחיישן המחבר לבקר

בחלק זה של הפעולות נדגים כיצד יוצריםلوح בקירה הכלול משק גրפי המציג טמפרטורה הנקלטת מחיישן המחבר לבקר ESP32.

ניצור תחילה Channel נוסף בשם חדש כדוגמת `esp32_new/Sensor`.
אפשרות נוספת היא לעדכן את ה- Channel שיצרנו עבור ה- LED ולהוסיף לו RESOURCE חדש. כמוzeigt באירוע:

בשלב הבא נוסיף לאותו לוח בקרה שיצרנו כבר בעבר ה- LED את הרכיב הבא:



נקבל לוח בקרה הכלול את הרכיבים הבאים:

בשלב הבא נכתוב את הקוד הבא:

```
from machine import Timer
from umqtt.simple import MQTTClient
import utime
import os
import sys

PUBLISH_PERIOD    = 10
msgNumber = 12.24
timeSave = 0

CHANNEL_TOKEN = 'token_jraXXXXXXXXXku'
CHANNEL_NAME   = 'esp32'
RESOURCE_NAME  = 'sensor'
MQTT_SERVER = 'mqtt.beebotte.com'
MQTT_USER = 'token:' + CHANNEL_TOKEN
MQTT_TOPIC = CHANNEL_NAME + '/' + RESOURCE_NAME

def handleTimerInt(timer):
    global msgNumber
    global timeSave
    msg = b'{"data": ' + str(msgNumber) + b', "write": true}'
    client.publish(MQTT_TOPIC,msg, qos=0)
    print("Publish:",msg)
    msgNumber += 1.12
    timeSave = 0

# create a random MQTT clientID
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_'+str(random_num), 'utf-8')

client = MQTTClient(mqtt_client_id, MQTT_SERVER, user=MQTT_USER,
password='', keepalive=PUBLISH_PERIOD*2)

myTimer = Timer(0)

try:
    client.connect()
    myTimer.init(period=PUBLISH_PERIOD*1000, mode=Timer.PERIODIC,
callback=handleTimerInt)
except Exception as e:
    print('could not connect to MQTT server {}'.format(type(e).__name__, e))
    sys.exit()

while True:
    try:
        print('Publish new data in', (PUBLISH_PERIOD-timeSave), "second.",
", end="\r")
        utime.sleep(1)
        timeSave+=1
    except KeyboardInterrupt:
        print('Ctrl-C pressed... exiting')
```

```

client.disconnect()
sys.exit()

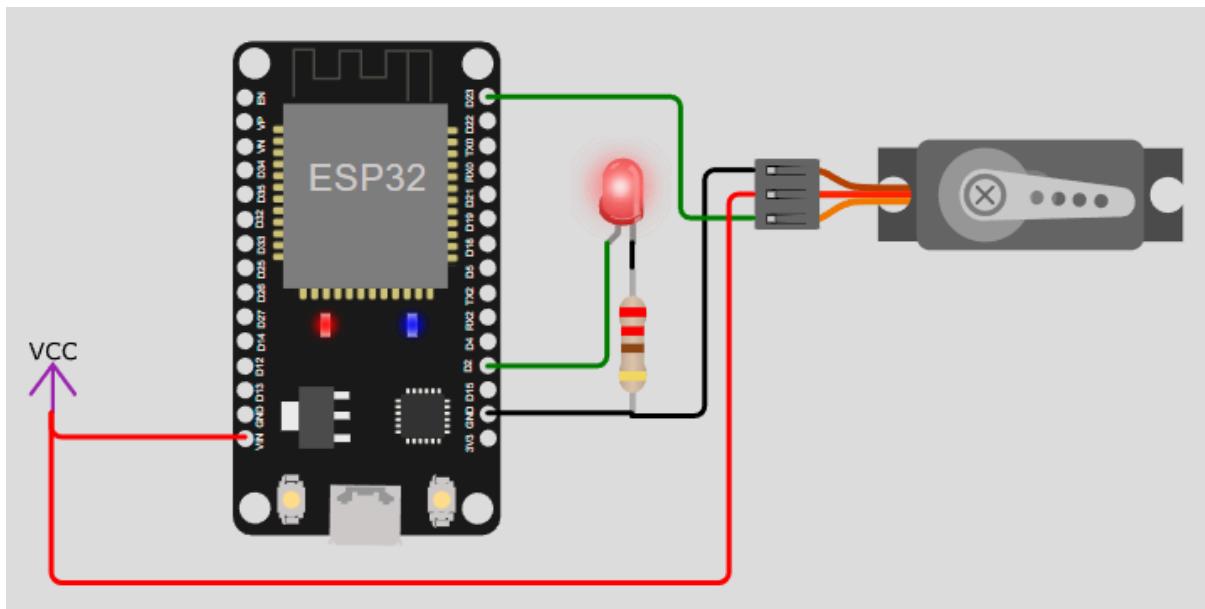
```

נקבל את הפלט הבא:

The screenshot shows the Beebotte interface. On the left, a terminal window titled 'Shell' displays MicroPython code running on an ESP32. The code includes a 'soft reboot' command and several 'Publish' statements to a topic, with the last one being 'esp32.sensor' at value '15.6'. A red arrow points from the terminal output to the corresponding entry in the dashboard on the right. The dashboard, titled 'My Dashboard', shows two channels: 'esp32.led' (status 'ON') and 'esp32.sensor' (value '15.60'). A second red arrow points from the 'esp32.sensor' entry in the dashboard back to the terminal output.

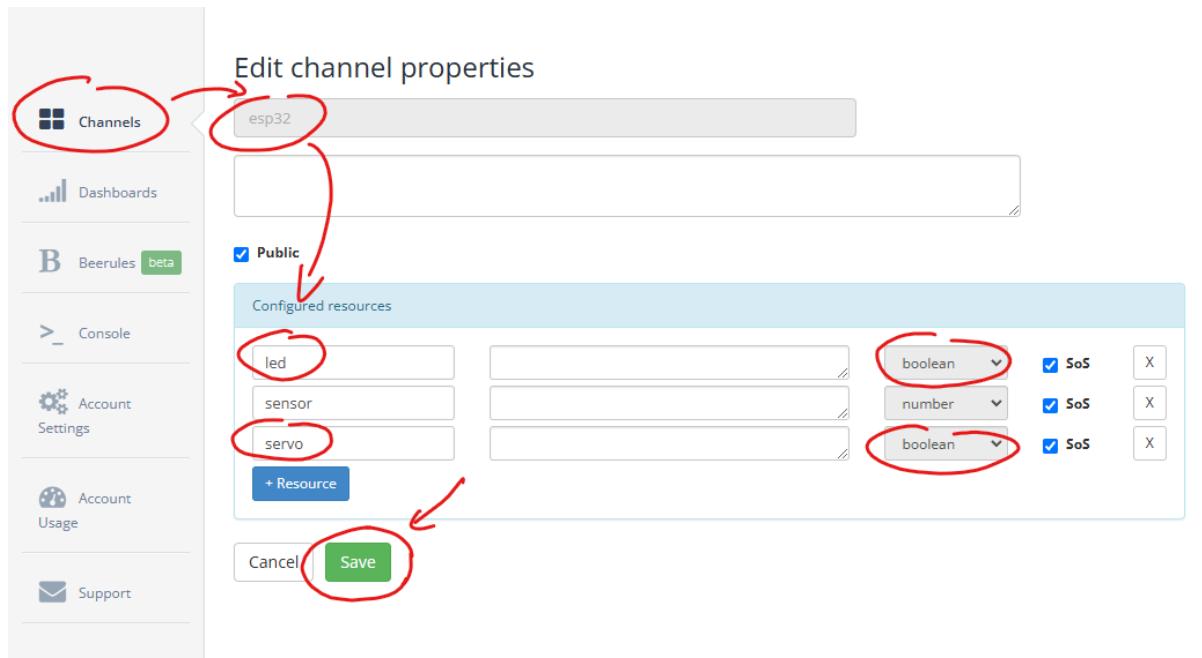
שימוש באתר beebotte.com במטרה לפרסם מספר הודעת במקביל לאותו הבקר

בחלק זה של הפעולות נדגים כיצד יוצרים לוח בקרה הכלול משיק גרפִי המציג שני לחצנים שבו כל אחד מהם מפעיל רכיבים שונים בחומרה. בדוגמה זו הרכיב הראשון יהיה מנוע סרוו והשני נורית LED. להלן שרטוט מעגל החומרה:

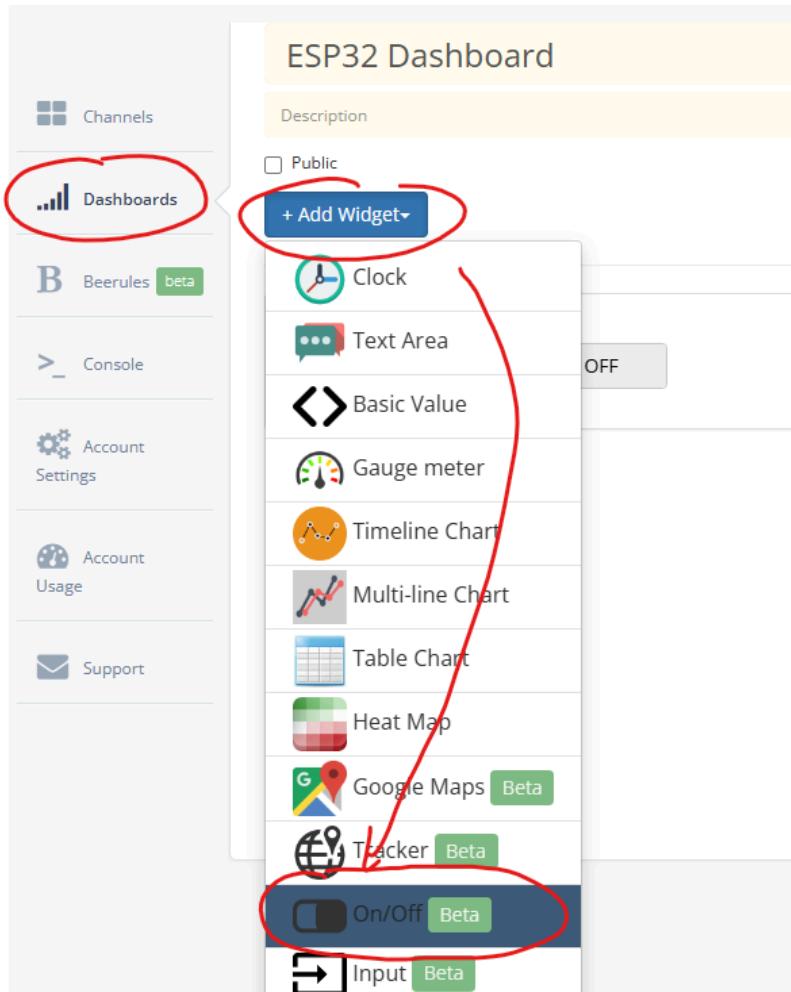


ניצור תחילה Channel נוסף בשם חדש כדוגמת `esp32_new_led` שני משאבים (Resources) האחד בשם led והשני servo.

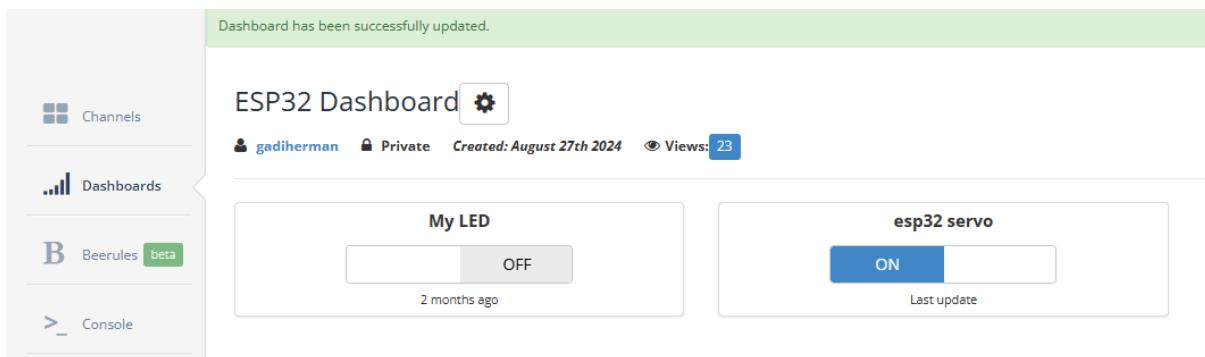
אפשרות נוספת היא לעדכן את ה- Channel שיצרנו עבור ה- LED ולהוסיף לו RESOURCE חדש. כמו צג באירוע:



בשלב הבא נוסיף לאותו לוח בקרה שיצרנו כבר בעבר ה- LED לחץ נוסף עליון הסרו:



נקלט לוח בקרה הכלל את הרכיבים הבאים:



בשלב הבא נכתוב את הקוד הבא:

```
#mqtt5.py
import time
from machine import Pin, PWM, Timer
from umqtt.simple import MQTTClient
import ujson
import os
import sys

# MQTT הגדרות
CHANNEL_TOKEN = 'token_jraXXXXXXXXXuku'
CHANNEL_NAME = 'esp32'
RESOURCE_NAME1 = 'servo'
RESOURCE_NAME2 = 'led'
MQTT_SERVER = 'mqtt.beebotte.com'
MQTT_USER = 'token:' + CHANNEL_TOKEN
MQTT_TOPIC1 = CHANNEL_NAME + '/' + RESOURCE_NAME1
MQTT_TOPIC2 = CHANNEL_NAME + '/' + RESOURCE_NAME2

# LED-1 פינום עבור סרבו
servo_pin = Pin(23, Pin.OUT)
led_pin = Pin(2, Pin.OUT)
pwm = PWM(servo_pin, freq=50)

# MQTT טוימר לשילוח פינג לשרת
PING_PERIOD = 120 # שניות
myTimer = Timer(0)

def handleTimerInt(timer):
    """שמירה על חיבור פעיל באמצעות פינג תקופתי"""
    try:
        client.ping()
        print('נשלח פינג')
    except Exception as e:
        print(f'שגיאת פינג: {e}')

def set_servo_angle(angle):
    """הגדרת זווית הסרבו בין 0 ל-180 מעלות"""
    try:
        # טיפוס SG90 המרת זווית לערך מדויר עבורה (50-110 מעלות)
        pass
    except:
        pass
```

```

        duty = int((angle / 180) * 75 + 40)
        pwm.duty(duty)
        print(f"{'מגילות':>20}-הסרו {angle}")
    except Exception as e:
        print(f"{'שגיאה':>20} {e}")

def mqtt_callback(topic, msg):
    """טיפול בהודעות נכנסות עבור הservo זה"""
    try:
        topic = topic.decode('utf-8')
        print(f"{'התקבל':>20} {topic}: {msg}")

        json_data = ujson.loads(msg)
        dt = json_data.get("data")

        if dt is None:
            print("{'אין':>20} שדה 'data' בהודעה")
            return

        if not isinstance(dt, bool):
            print("{'שדה':>20} 'data' שהתקבל אינו ערך בוליאני")
            return

        if topic == MQTT_TOPIC1: # בקרת servo
            set_servo_angle(90 if dt else 0)
        elif topic == MQTT_TOPIC2: # LED
            led_pin.value(1 if dt else 0)
            print(f"{'LED':>20} {dt ? 'דולק' : 'כבוי' }")

    except Exception as e:
        print(f"{'שגיאה':>20} בעיה בהודעה {e}")

# MQTT ללקוח עם מידע אكريאי
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_' + str(random_num), 'utf-8')

# MQTT והגדרת לקוח
client = MQTTClient(
    mqtt_client_id,
    MQTT_SERVER,
    user=MQTT_USER,
    password='',
    keepalive=PING_PERIOD * 2
)

# התחברות והרשמה לנושאים
try:
    client.connect()
    myTimer.init(period=PING_PERIOD * 1000, mode=Timer.PERIODIC,
    callback=handleTimerInt)

    # והרשמה לשני הנושאים הגדרת פונקציית
    client.set_callback(mqtt_callback)
    client.subscribe(MQTT_TOPIC1)

```

```
client.subscribe(MQTT_TOPIC2)

print("MQTT מתחבר לברוקר")

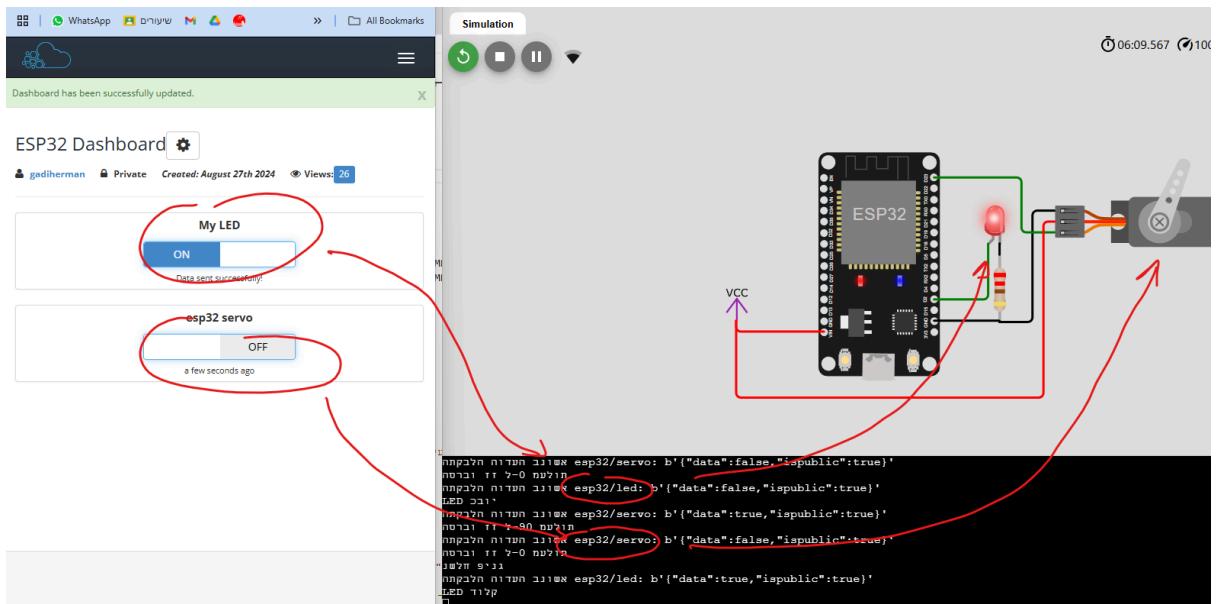
# ראשית לולאה
while True:
    # בדיקת הודעות ללא חסימה
    client.check_msg()
    # צפופה לולאה קלה
    time.sleep(0.1)
    # השהייה קצרה למניעת עכבות

except Exception as e:
    print(f"MQTT שגיאת: {type(e).__name__} {e}")
    sys.exit(1)

finally:
    try:
        client.disconnect()
        pwm.deinit()
    except:
        pass
```

* תודה לשלי מוסינקו על הקוד.

נקבל את הפלט הבא:



כיתן להתחבר לאתר הבא כדי להפעיל סימולציה מלאה של ביסוי זה. להלן קישור לSIMOLICA:

<https://wokwi.com/projects/359801682833812481>

רישום בקר ESP32 לקבלת מידע מבקר אחר

כפי שראינו ב-2 הדוגמאות הקודמות בAKER ESP32 מסוגל להירשם לשירות קבלת נתונים. לדוגמה לקבל מידע מלחץן המשמש במכשיר שבקרה. כמו כן ראיינו שבAKER ESP32 מסוגל גם לספק מידע מעורר אחרים שנרשמו למידע. לדוגמה ראיינו קוד המספק למכשיר המשמש שבקרה מידע על טמפרטורה.

ניתן לרשום בקר אחד שיספק את המידע ובקר אחר יקבל את המידע (כMOVED דרך שירות הענן).

הקוד שמספק את המידע כבר הוגם בדוגמה השנייה. להלן דוגמה לתוכנה בברker נוספת הנרשם לקבל מידע מהברker הראשון. במצב זה נקבל מצב שבוiker אחד מספק לענן מידע על הדלקה וכיבוי נורית הלהד בקר שני מקבל את המידע על הנורית מהענן ומודליק LED. להלן הקוד:

```
#mqtt6.py
from machine import Timer
from umqtt.simple import MQTTClient
import utime
import os
import sys

PUBLISH_PERIOD    = 10
Led = True
timeSave = 0

CHANNEL_TOKEN = 'token_jraXXXXXXXXXUku'
CHANNEL_NAME   = 'esp32'
RESOURCE_NAME  = 'led'
MQTT_SERVER = 'mqtt.beebotte.com'
MQTT_USER = 'token:' + CHANNEL_TOKEN
MQTT_TOPIC = CHANNEL_NAME + '/' + RESOURCE_NAME

def handleTimerInt(timer):
    global Led
    global timeSave
    msg1 = b'{"data":true,"ispublic":false}'
    msg2 = b'{"data":false,"ispublic":false}'
    if Led:
        client.publish(MQTT_TOPIC,msg1, qos=0)
        print("Publish:",msg1)
    else:
        client.publish(MQTT_TOPIC,msg2, qos=0)
        print("Publish:",msg2)
    Led = not Led
    timeSave = 0

# create a random MQTT clientID
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_'+str(random_num), 'utf-8')

client = MQTTClient(mqtt_client_id, MQTT_SERVER, user=MQTT_USER,
password='', keepalive=PUBLISH_PERIOD*2)

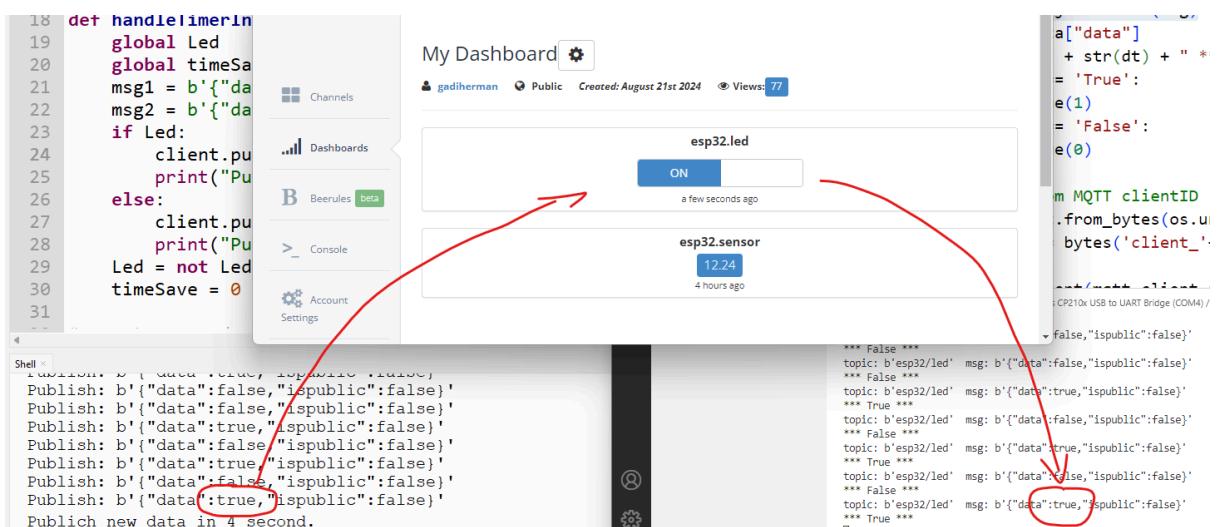
myTimer = Timer(0)

try:
    client.connect()
    myTimer.init(period=PUBLISH_PERIOD*1000, mode=Timer.PERIODIC,
callback=handleTimerInt)
except Exception as e:
    print('could not connect to MQTT server {}{}'.format(type(e).__name__,
e))
```

```
    sys.exit()

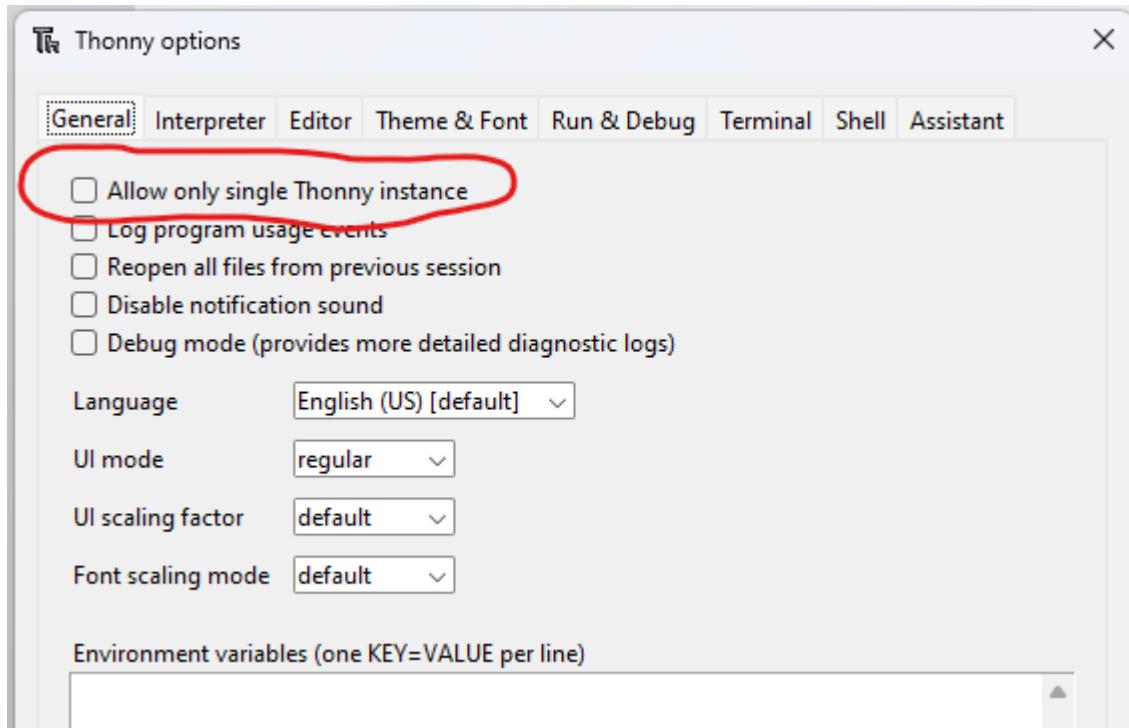
while True:
    try:
        print('Publish new data in', (PUBLISH_PERIOD-timeSave) , "second."
", end="\r")
        utime.sleep(1)
        timeSave+=1
    except KeyboardInterrupt:
        print('Ctrl-C pressed...exiting')
        client.disconnect()
        sys.exit()
```

נקבל את הפלט הבא:



טיג!

כדי לפתח מספר חלונות של סביבת הפייטו thonny שבעל חלון נעבד מול בקר ESP32 אחר. יש ללקת להגדרות של סביבת העבודה "General" => Options => Tools ובדל את הסימון כפי שמופיע באירור הבא:



כתיבת קוד לבקר ESP32 ועובד ב- `duplex` (מפרסם ומתקבל מיידע בו זמנית)

כפי שראינו ב-3 הדוגמאות הקודמות בקיר ESP32 מסוגל להירשם לשירותים קבלת נתונים. לדוגמה לקבל מידע מלחץ הממוקם במשתק המשמש שבאטר. כמו כן שבקיר ESP32 מסוגל גם לספק מידע עבור אחרים שנרשמו למידע. לדוגמה ראיינו קוד המספק למשתק המשמש שבאטר מייד על טמפרטורה.

בדוגמה שלහלן נראה כיצד לכתוב קוד המפרסם מייד על טמפרטורה ובו זמנית נרשם לשירות הגורם להפעלת נורית ה-LED. להלן הקוד:

```
#mqtt7.py
from machine import Pin, Timer
from umqtt.simple import MQTTClient
import ujson
import sys
import os
from time import sleep

LED = Pin(2, Pin.OUT)
PING_PERIOD = 120

PUBLISH_PERIOD = 10
msgNumber = 12.24

CHANNEL_TOKEN = 'token_jXXXXXXXXXXXXXXXXXXXX'
CHANNEL_NAME = 'esp32'
MQTT_SERVER = 'mqtt.beebotte.com'
MQTT_USER = 'token:' + CHANNEL_TOKEN
```

```

RESOURCE_NAME_PUBLISH = 'sensor'
RESOURCE_NAME_SUBSCRIBE = 'led'
MQTT_TOPIC_PUBLISH = CHANNEL_NAME + '/' + RESOURCE_NAME_PUBLISH
MQTT_TOPIC_SUBSCRIBE = CHANNEL_NAME + '/' + RESOURCE_NAME_SUBSCRIBE

def handleTimer0Int(timer):
    client.ping()
    print('ping')

def handleTimer1Int(timer):
    global msgNumber
    global timeSave
    msg = b'{"data": ' + str(msgNumber) + b', "write": true}'
    client.publish(MQTT_TOPIC_PUBLISH, msg, qos=0)
    print("Publish:", msg)
    msgNumber += 1.12
    timeSave = 0

def callback_func(topic, msg):
    print("topic:", topic, " msg:", msg)
    json_data = ujson.loads(msg)
    dt = json_data["data"]
    print("*** " + str(dt) + " ***")
    if dt:
        LED.value(1)
    else:
        LED.value(0)

# create a random MQTT clientID
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_' + str(random_num), 'utf-8')

client = MQTTClient(mqtt_client_id, MQTT_SERVER, user=MQTT_USER,
password='', keepalive=PING_PERIOD*2)

timer0 = Timer(0) #subscribe
timer1 = Timer(1) #publish

try:
    client.connect()
    timer0.init(period=PING_PERIOD*1000, mode=Timer.PERIODIC,
callback=handleTimer0Int)
    timer1.init(period=PUBLISH_PERIOD*1000, mode=Timer.PERIODIC,
callback=handleTimer1Int)
except Exception as e:
    print('could not connect to MQTT server {}'.format(type(e).__name__,
e))
    sys.exit()

client.set_callback(callback_func)
client.subscribe(MQTT_TOPIC_SUBSCRIBE)

while True:
    try:

```

```

        client.wait_msg()
    except KeyboardInterrupt:
        print('Ctrl-C pressed... exiting')
        client.disconnect()
        sys.exit()

```

נקבל את הפלט הבא:

mqtt_4_duplex.py

```

31 msg = b'{"data": ' + str(msgNumber) + b'}', qos=0
32 client.publish(MQTT_TOPIC_PUBLISH, msg, qos=0)
33 print("Publish:", msg)
34 msgNumber += 1.12
35 timeSave = 0
36
37 def callback_func(topic, msg):
38     print("topic:", topic, "msg:", msg)
39     json_data = ujson.loads(msg)
40     dt = json_data["data"]
41     print("**** " + str(dt) + " ****")
42     if dt:
43         LED.value(1)
44     else:
45         LED.value(0)
46
47 # create a random MQTT clientID
48 random_num = int.from_bytes(os.urandom(3), 'little')

Shell >
7273, "ispublic": true}
*** True ***
Publish: b'{"data": 12.24, "write": true}'
topic: b'esp32/led'  msg: b'{"data": false, "ispublic": true}'
*** False ***
topic: b'esp32/led'  msg: b'{"data": true, "ispublic": true}'
*** True ***
Publish: b'{"data": 13.36, "write": true}'
Publish: b'{"data": 14.48, "write": true}'
```

Beebotte - gadiherman MicroPython_esp32_V3 - 100% beebotte.com/dash/5cb9d1a0-5fd1-11ef-a733-f904bc3a509

My Dashboard

Channels Dashboards Beerules Account Settings Account Usage Support

esp32.led ON a few seconds ago

esp32.sensor 14.48 a few seconds ago

ניתן לראות כיצד אותו קוד מאפשר גם לhirestem subscribe לשירות esp32.led ובאותו הזמן הקוד מספק כל 10
שניות פורסום publish לשירותים esp32.sensor.

שימוש לב שני השירותים יושבים על אותו הערוץ CHANNEL. כך:

esp32

gadiherman Public Created: August 24th 2024

Channel Token: tc

Configured resources		
led	true	5 minutes ago
sensor	46.96	a few seconds ago

יצירת חשבון באתר io.adafruit.com

IO הוא שירות ענן המיעוד לפרויקטים של אינטרנט של הדברים (IoT). להלן סקירה כללית על השירות שמציע האתר:

- מאפשר לחבר מכשיר IoT לענן ולנהל אותו מרוחוק.
- תומך בMagnitude רחב של התקנים חומרה, במיוחד מוצרים של Adafruit.
- מאפשר ליצור ממשק משתמש מותאם אישית לשיליטה וניהול של מכשיר IoT. כולל אלמנטים כמו כפתורים, מד מחוג, גրפים ועוד.
- מאפשר לאחסן ולנהל נתונים מחיי-שנים או מכשירים.
- ניתן להציג את הנתונים בצורה גרפית או ליצא אותם.
- מספק ברורiot MQTT לתקשורת עיליה בין מכשירים.
- מאפשר להציג פעולות אוטומטיות בהתאם מסוימים. לדוגמה, שליחת התראה כאשר טמפרטורה עולה מעל סף מסוים.
- מציע תוכנית חינמית עם מגבלות מסוימות, ותוכניות בתשלומים עם יכולות נרחבות יותר.

נתחבר לאתר הבא ונפתח בו חשבון משתמש:

<https://io.adafuit.com/>

לאחר התחברות מוצלחת נקבל את המסר הבא:

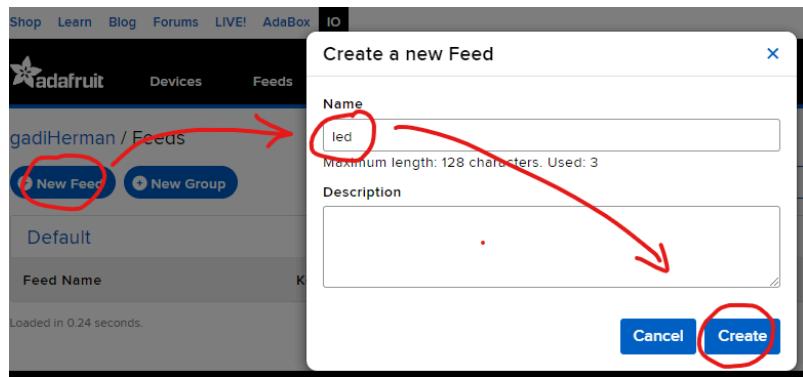
The screenshot shows the Adafruit IO dashboard for user 'gadiHerman'. At the top, there's a navigation bar with links to Shop, Learn, Blog, Forums, LIVE!, AdaBox, and IO. The IO link is highlighted. On the right, it says 'Hi, Gadi Herman | Account' and shows a shopping cart icon with '0' items. Below the navigation is a black header with tabs for Devices, Feeds, Dashboards, Actions, and Power-Ups. A 'New Device' button is on the right. The main area starts with a green banner that says: 'You are currently using a Adafruit IO Basic plan. For just \$10/month, upgrade to IO+ to unlock unlimited devices, groups, feeds, dashboards, and more! Learn about the other features and benefits of upgrading your account here.' Below the banner are two sections: 'Account Status' and 'Live Errors'. 'Account Status' shows metrics for Devices (0 of 2), Groups (1 of 5), Feeds (2 of 10), Dashboards (1 of 5), and Data Rate (0 of 30). 'Live Errors' shows 'No errors since page load.' Underneath is a section for 'My Dashboards'.

שימוש באתר io.adafuit.com המטרת לשלוט על רכיב חומרה

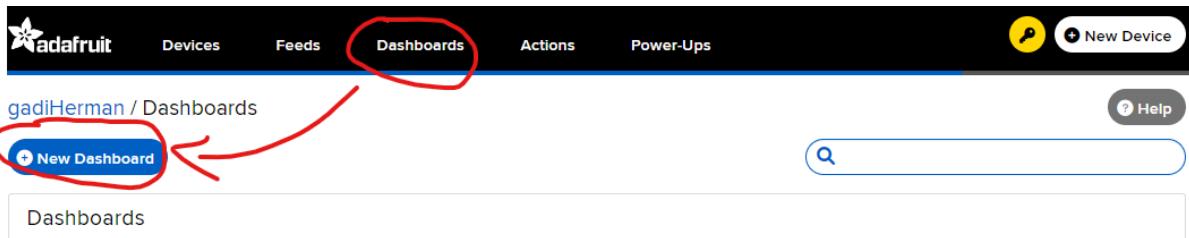
בחלק זה של הפעילות נדגים כיצד יוצריםلوح בקרה הכלל לחצן שמדליק ומכבה LED בחומרה.

יצור תחילה Feed חדש בשם led

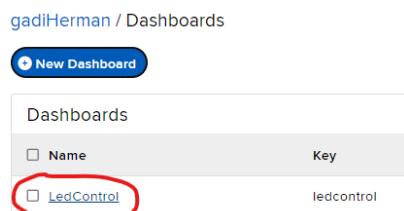
The screenshot shows the 'Feeds' section of the Adafruit IO dashboard. The 'Feeds' tab is highlighted with a red circle. Below it, there are two buttons: 'New Feed' and 'New Group', both circled in red. The main area shows a table with one row for 'Default'. The columns are 'Feed Name' (containing 'led'), 'Key' (empty), 'Last value' (empty), and 'Recorded' (empty). A search bar is at the top right, and a progress bar at the bottom says 'Loaded in 0.24 seconds.'



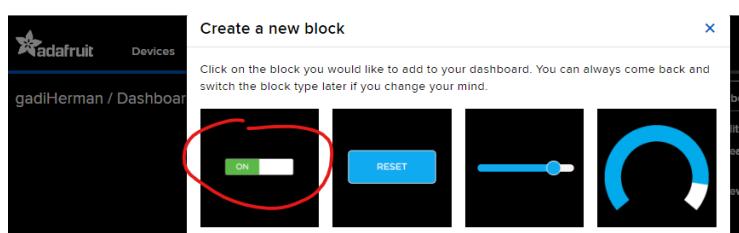
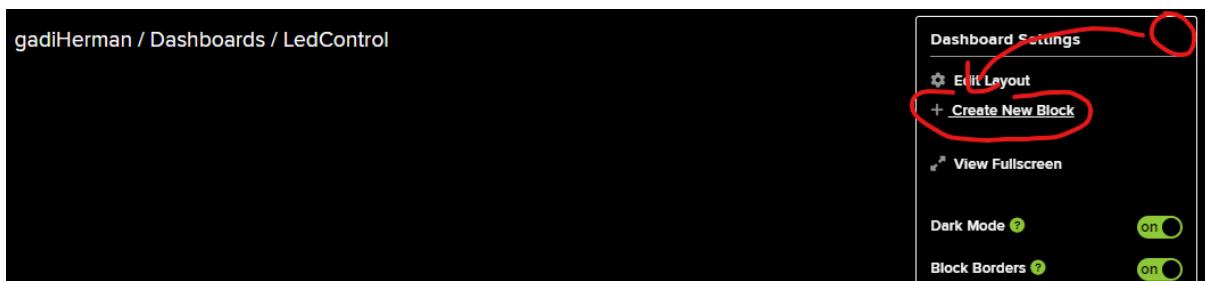
בשלב הבא ניצור לוח בקרה חדש בשם LedControl



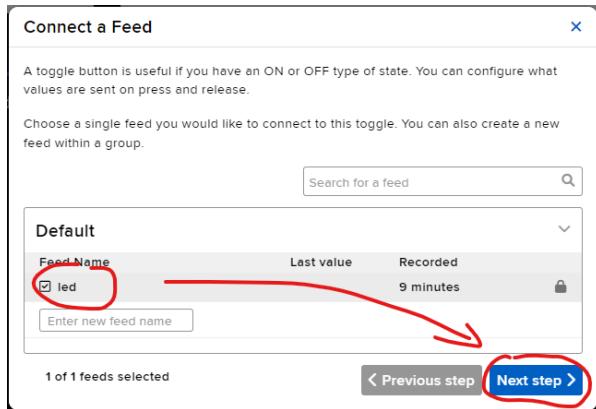
נכנו לעיצוב הממשק על ידי לחיצה של שמו:



נוסף בו לחץ על ידי לחיצה על :



בשלב הבא נקשר בין הלחצן שייצרנו עם ה- Feeds ששמו led שייצרנו מוקדם:



נגידר בו את המאפיינים הבאים:

Block Title (optional)

Button On Text

Limit of 6 characters for the toggle text. Use the block title to be more descriptive.

Button On Value (uses On Text if blank)

Button Off Text

Limit of 6 characters for the toggle text. Use the block title to be more descriptive.

Button Off Value (uses Off Text if blank)

Block Preview

ESP32 LED Control

OFF

Toggle A toggle button is useful if you have an ON or OFF type of state. You can configure what values are sent on press and release.

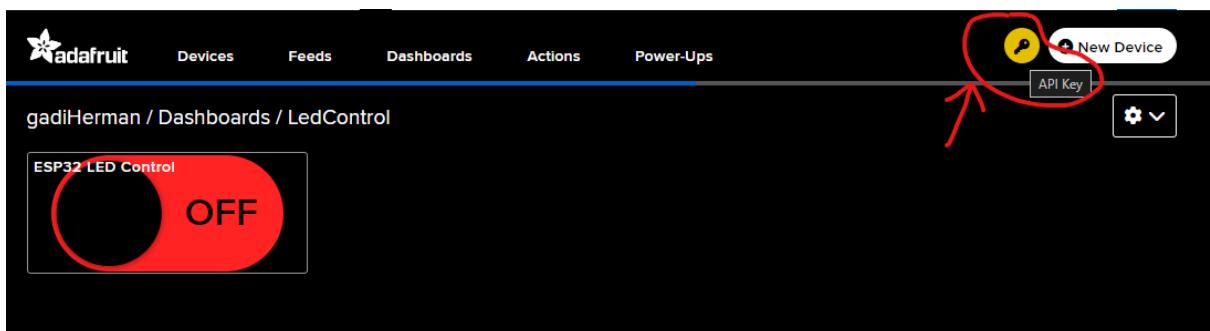
Test Value

Published Value

0 bytes

[< Previous step](#) **Create block!** [Next step >](#)

הדבר האחרון שהוא נדרש לו כדי לחבר בין האתר שמספק לנו שירות ענן לבין הילד שמחובר לבקר ESP32 הוא קוד הגישה הייחודי לכל משתמש. נקבל אותו על ידי לחיצה על המפתח:



נפתח את סביבת התכנות כדי לכתוב את הקוד הבא:

הקובץ boot.py

```
# This file is executed on every boot (including wake-boot from deepsleep)
import network

def connect():
    ssid = "XXXX"
    password = "XXXX"

    station = network.WLAN(network.STA_IF)

    if station.isconnected() == True:
        print("Already connected")
        return

    station.active(True)
    station.connect(ssid, password)

    while station.isconnected() == False:
        pass

    print("Connection successful")
    print(station.ifconfig())

connect()
```

הקובץ main.py

```
import time
from umqtt.robust import MQTTClient
import os
import sys

# the following function is the callback which is
# called when subscribed data is received
def cb(topic, msg):
    print('Received Data: Topic = {}, Msg = {}'.format(topic, msg))

# create a random MQTT clientID
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_'+str(random_num), 'utf-8')

# connect to Adafruit IO MQTT broker using unsecure TCP (port 1883)
ADAFRUIT_IO_URL = b'io.adafruit.com'
ADAFRUIT_USERNAME = b'gadiHerman'
ADAFRUIT_IO_KEY = b'aio__XXX_p31'
ADAFRUIT_IO_FEEDNAME = b'led'

client = MQTTClient(client_id= mqtt_client_id,
                    server=ADAFRUIT_IO_URL,
                    user=ADAFRUIT_USERNAME,
                    password=ADAFRUIT_IO_KEY,
```

```

        ssl=False)

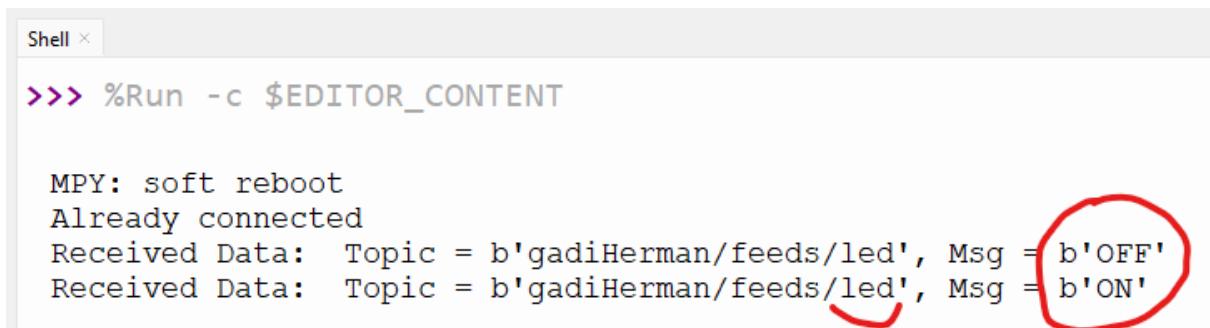
try:
    client.connect()
except Exception as e:
    print('could not connect to MQTT server {}{}'.format(type(e).__name__,
e))
    sys.exit()

mqtt_feedname = bytes('{:s}/feeds/{:s}'.format(ADAFRUIT_USERNAME,
ADAFRUIT_IO_FEEDNAME), 'utf-8')
client.set_callback(cb)
client.subscribe(mqtt_feedname)

# wait until data has been Published to the Adafruit IO feed
while True:
    try:
        client.wait_msg()
    except KeyboardInterrupt:
        print('Ctrl-C pressed... exiting')
        client.disconnect()
        sys.exit()

```

לאחר צריבת הרכיב וחיבורו לאינטרנט על ידי תקשורת WiFi בכל פעם שנלחץ על הלחצן שבאerator נקלט בברך את ההודעה הבא:



```

Shell >>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Already connected
Received Data: Topic = b'gadiHerman/feeds/led', Msg = b'OFF'
Received Data: Topic = b'gadiHerman/feeds/led', Msg = b'ON'

```

כל מה שנותר לעשות זה להוסיף פעולה cb את הקוד הבא כדי להפעיל בפועל את ה- LED בהתאם לפקודה שקיבל. נדגים זאת:

```

import time
from umqtt.robust import MQTTClient
import os
import sys
from machine import Pin
Led = Pin(2, mode=Pin.OUT, value=0) # 0V on output

# the following function is the callback which is
# called when subscribed data is received
def cb(topic, msg):
    print('Received Data: Topic = {}, Msg = {}'.format(topic, msg))
    if msg==b'OFF':
        Led.off()
    if msg==b'ON':

```

```

    Led.on()

# create a random MQTT clientID
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_'+str(random_num), 'utf-8')

# connect to Adafruit IO MQTT broker using unsecure TCP (port 1883)
ADAFRUIT_IO_URL = b'io.adafruit.com'
ADAFRUIT_USERNAME = b'gadiHerman'
ADAFRUIT_IO_KEY = b'aio_____p31'
ADAFRUIT_IO_FEEDNAME = b'led'

client = MQTTClient(client_id= mqtt_client_id,
                     server=ADAFRUIT_IO_URL,
                     user=ADAFRUIT_USERNAME,
                     password=ADAFRUIT_IO_KEY,
                     ssl=False)

try:
    client.connect()
except Exception as e:
    print('could not connect to MQTT server {}{}'.format(type(e).__name__, e))
    sys.exit()

mqtt_feedname = bytes('{:s}/feeds/{:s}'.format(ADAFRUIT_USERNAME,
ADAFRUIT_IO_FEEDNAME), 'utf-8')
client.set_callback(cb)
client.subscribe(mqtt_feedname)

# following two lines is an Adafruit-specific implementation of the Publish
"retain" feature
# which allows a Subscription to immediately receive the last Published
value for a feed,
mqtt_feedname_get = bytes('{:s}/get'.format(mqtt_feedname), 'utf-8')
client.publish(mqtt_feedname_get, '\0')

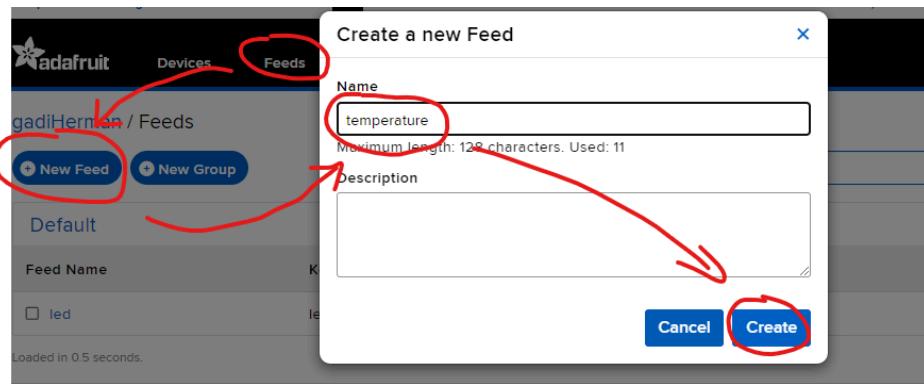
# wait until data has been Published to the Adafruit IO feed
while True:
    try:
        client.wait_msg()
    except KeyboardInterrupt:
        print('Ctrl-C pressed... exiting')
        client.disconnect()
        sys.exit()

```

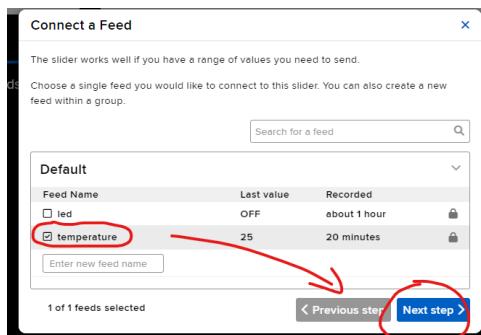
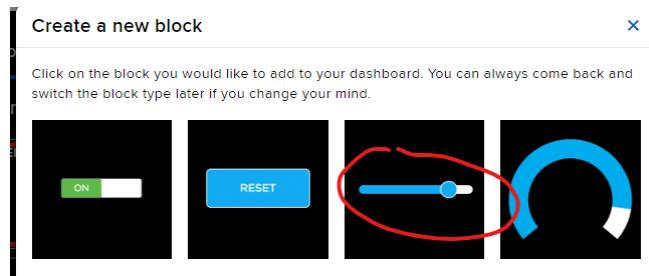
שימוש באתר io.adafruit.com המטרה לקלוט מידע מיידי מהחומר לבקר

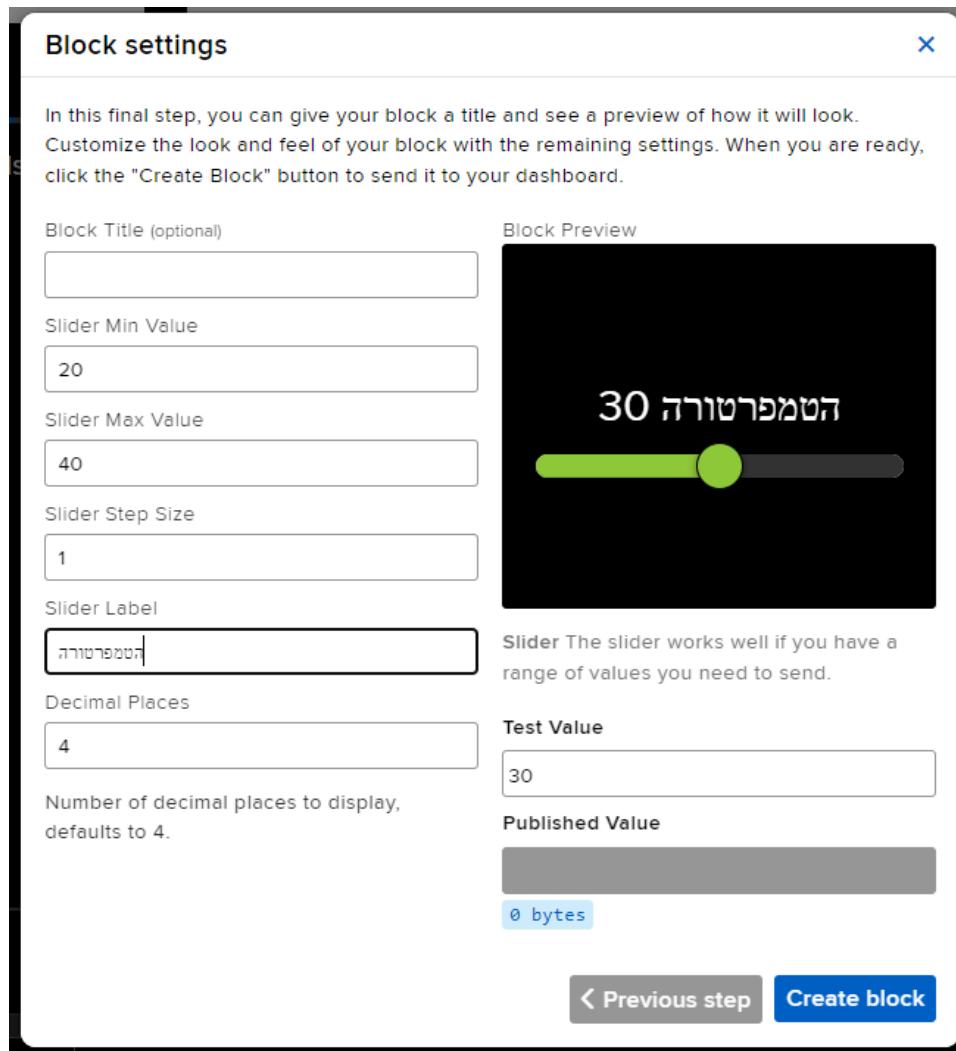
ב חלק זה של הפעילותות נדגים כיצד יוצריםلوح בקרה הכלל משק גרפי המציג טמפרטורה הנקלטת מחיישן המחבר לבקר ESP32.

ניצור תחילה Feeds נוסף בשם temperature

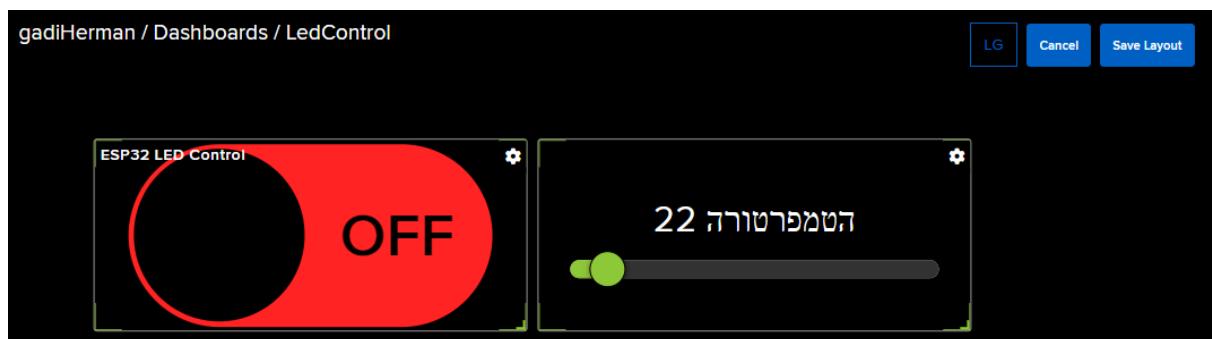


בשלב הבא נוסיף לאותו לוח בקרה שיצרנו כבר בעבר ה- LED את הרכיב הבא:





נקבל לוח בקרה הכלול את הרכיבים הבאים:



בשלב הבא נכתוב את הקוד הבא:

```
import time
from umqtt.robust import MQTTClient
import os
import sys

# create a random MQTT clientID
```

```

random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_'+str(random_num), 'utf-8')

# connect to Adafruit IO MQTT broker using unsecure TCP (port 1883)
#
# To use a secure connection (encrypted) with TLS:
#   set MQTTClient initializer parameter to "ssl=True"
#   Caveat: a secure connection uses about 9k bytes of the heap
#           (about 1/4 of the micropython heap on the ESP8266 platform)
ADAFRUIT_IO_URL = b'io.adafruit.com'
ADAFRUIT_USERNAME = b'gadiHerman'
ADAFRUIT_IO_KEY = b'aio_____p31'
ADAFRUIT_IO_FEEDNAME = b'temperature'

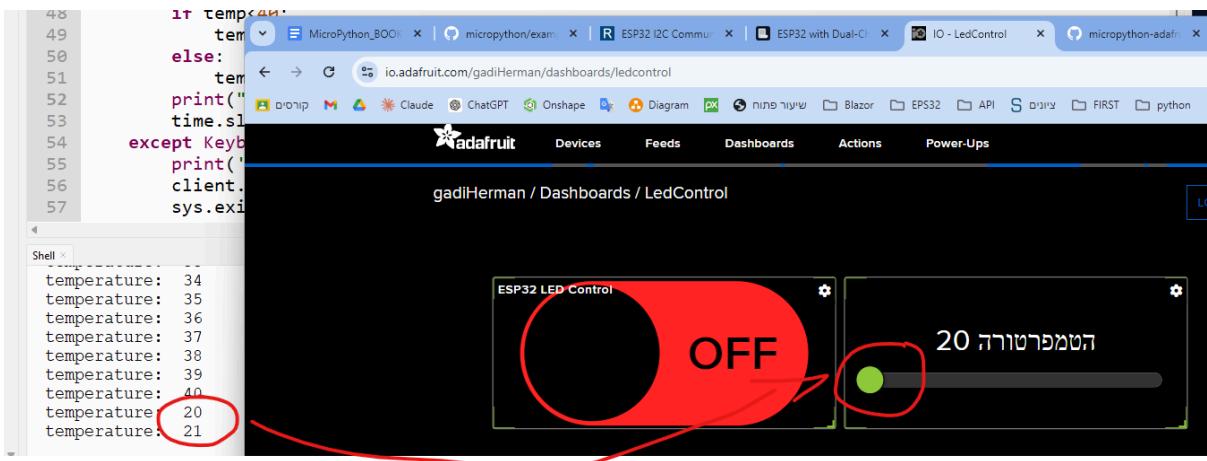
client = MQTTClient(client_id=mqtt_client_id,
                     server=ADAFRUIT_IO_URL,
                     user=ADAFRUIT_USERNAME,
                     password=ADAFRUIT_IO_KEY,
                     ssl=False)

try:
    client.connect()
except Exception as e:
    print('could not connect to MQTT server {}'.format(type(e).__name__, e))
    sys.exit()

# publish free heap statistics to Adafruit IO using MQTT
#
# format of feed name:
#   "ADAFRUIT_USERNAME/feeds/ADAFRUIT_IO_FEEDNAME"
mqtt_feedname = bytes('{:s}/feeds/{:s}'.format(ADAFRUIT_USERNAME,
ADAFRUIT_IO_FEEDNAME), 'utf-8')
PUBLISH_PERIOD_IN_SEC = 10
temp=20
while True:
    try:
        client.publish(mqtt_feedname,
                       bytes(str(temp), 'utf-8'),
                       qos=0)
        if temp<40:
            temp+=1
        else:
            temp=20
        print("temperature: ", temp)
        time.sleep(PUBLISH_PERIOD_IN_SEC)
    except KeyboardInterrupt:
        print('Ctrl-C pressed... exiting')
        client.disconnect()
        sys.exit()

```

נקבל את הפלט הבא:



רישום בקר לקלט מידע מברך אחר

כפי שראינו ב-2 הדוגמאות הקודומות בקר ESP32 מסוגל להירשם לשירות קבלת נתונים. לדוגמה לקבל מידע מלחצן הממוקם במכשיר המשמש שבatter. כמו כן ראיינו שבקר ESP32 מסוגל גם לספק מידע עבור אחרים שנרשמו למידע. לדוגמה ראיינו קוד המספק למכשיר המשמש שבatter מידע על טמפרטורה.

ניתן כמובן לרשום בקר אחד שיספק את המידע ובקר אחר יקבל את המידע (כמובן דרך שירות הענן).

הקוד שמספק את המידע כבר הודגם בדוגמה השנייה. להלן דוגמה לתוכנה בברך נוסף הנרשם לקבל מידע מהברך הראשון. במצב זה נקלט מצב שבברך אחד מספק לענן מידע עם טמפרטורה ובברך שני מקבל את המידע על הטמפרטורה מהענן ומדליק LED או מערכת קירור אחרת במידה והטמפרטורה גבוהה מ- 30 מעלות. להלן :

```
import time
from umqtt.robust import MQTTClient
import os
import sys
from machine import Pin
Led = Pin(2, mode=Pin.OUT, value=0)

# the following function is the callback which is
# called when subscribed data is received
def cb(topic, msg):
    print('Received Data: Topic = {}, Msg = {}'.format(topic, msg))
    if int(msg) < 30:
        Led.off()
    else:
        Led.on()

# create a random MQTT clientID
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_'+str(random_num), 'utf-8')

# connect to Adafruit IO MQTT broker using unsecure TCP (port 1883)
ADAFRUIT_IO_URL = b'io.adafruit.com'
ADAFRUIT_USERNAME = b'gadiHerman'
ADAFRUIT_IO_KEY = b'aio_UKmN85DbcrlQpriPBAmRTgbm0p31'
ADAFRUIT_IO_FEEDNAME = b'temperature'
```

```

client = MQTTClient(client_id= mqtt_client_id,
                     server=ADAFRUIT_IO_URL,
                     user=ADAFRUIT_USERNAME,
                     password=ADAFRUIT_IO_KEY,
                     ssl=False)

try:
    client.connect()
except Exception as e:
    print('could not connect to MQTT server {}{}'.format(type(e).__name__, e))
    sys.exit()

mqtt_feedname = bytes('{:s}/feeds/{:s}'.format(ADAFRUIT_USERNAME, ADAFRUIT_IO_FEEDNAME), 'utf-8')
client.set_callback(cb)
client.subscribe(mqtt_feedname)

# wait until data has been Published to the Adafruit IO feed
while True:
    try:
        client.wait_msg()
    except KeyboardInterrupt:
        print('Ctrl-C pressed... exiting')
        client.disconnect()
        sys.exit()

```

בקר ESP32 שגם מספק מידע וגם מקבל מידע בו בזמןית

להלן דוגמת קוד המשלבת פונקציית publish וsubscribe באותו הזמן:

```

import time
from umqtt.robust import MQTTClient
import os
import sys
from machine import Pin
Led = Pin(2, mode=Pin.OUT, value=0) # 0V on output

# the following function is the callback which is
# called when subscribed data is received
def cb(topic, msg):
    print('Subscribe: Received Data: Topic = {}, Msg = {}{}'.format(topic, msg))
    if msg==b'OFF':
        Led.off()
    if msg==b'ON':
        Led.on()

# create a random MQTT clientID
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_'+str(random_num), 'utf-8')

```

```

ADAFRUIT_IO_URL = b'io.adafruit.com'
ADAFRUIT_USERNAME = b'gadiHerman'
ADAFRUIT_IO_KEY = b'aio_XXX_0p3l'
ADAFRUIT_IO_SUB_FEEDNAME = b'led'
ADAFRUIT_IO_PUB_FEEDNAME = b'temperature'

client = MQTTClient(client_id= mqtt_client_id,
                     server=ADAFRUIT_IO_URL,
                     user=ADAFRUIT_USERNAME,
                     password=ADAFRUIT_IO_KEY,
                     ssl=False)

try:
    client.connect()
except Exception as e:
    print('could not connect to MQTT server {}{}'.format(type(e).__name__, e))
    sys.exit()

# format of feed name: "ADAFRUIT_USERNAME/feeds/ADAFRUIT_IO_FEEDNAME"
mqtt_pub_feedname = bytes('{:s}/feeds/{:s}'.format(ADAFRUIT_USERNAME,
ADAFRUIT_IO_PUB_FEEDNAME), 'utf-8')
mqtt_sub_feedname = bytes('{:s}/feeds/{:s}'.format(ADAFRUIT_USERNAME,
ADAFRUIT_IO_SUB_FEEDNAME), 'utf-8')
client.set_callback(cb)
client.subscribe(mqtt_sub_feedname)
PUBLISH_PERIOD_IN_SEC = 10
SUBSCRIBE_CHECK_PERIOD_IN_SEC = 0.5
accum_time = 0
temp=20
while True:
    try:
        # Publish
        if accum_time >= PUBLISH_PERIOD_IN_SEC:
            print('Publish: temperature = {}'.format(temp))
            client.publish(mqtt_pub_feedname,
                            bytes(str(temp), 'utf-8'),
                            qos=0)
        accum_time = 0
        if temp<40:
            temp+=1
        else:
            temp=20
        # Subscribe. Non-blocking check for a new message.
        client.check_msg()

        time.sleep(SUBSCRIBE_CHECK_PERIOD_IN_SEC)
        accum_time += SUBSCRIBE_CHECK_PERIOD_IN_SEC
    except KeyboardInterrupt:
        print('Ctrl-C pressed... exiting')
        client.disconnect()
        sys.exit()

```

נקבל את הפלט הבא:

The screenshot shows the Adafruit IO dashboard interface. At the top, there's a navigation bar with links for Devices, Feeds, Dashboards, Actions, and Power-Ups. Below the navigation bar, the URL gadiHerman / Dashboards / LedControl is displayed. The main area of the dashboard features two components: a red circular button labeled "ESP32 LED Control" with the word "OFF" in white, and a slider gauge showing a value of 33 with the text "הטמפרטורה" above it. At the bottom of the screen, there's a terminal window titled "Shell" showing the following log output:

```
Shell x
Publish: temperature = 25
Publish: temperature = 26
Publish: temperature = 27
Publish: temperature = 28
Publish: temperature = 29
Subscribe: Received Data: Topic = b'gadiHerman/feeds/led', Msg = b'OFF'
Subscribe: Received Data: Topic = b'gadiHerman/feeds/led', Msg = b'ON'
Publish: temperature = 30
Subscribe: Received Data: Topic = b'gadiHerman/feeds/led', Msg = b'OFF'
Publish: temperature = 31
Publish: temperature = 32
Publish: temperature = 33
```

שימוש באתר beebotte.com במטרה לשדר לו מחרוזות String

בחלק זה של הפעילות נדגים כיצד יוצרים לוח בקרה הכלל ממשק גרפי המציג מחרוזות טקסט הנקלטת מalker. ArdDioainer המחבר בתקשורת טורית לבקר ESP32.
יצור תחילה Channel חדש בשם esp32/getstring .
אפשרות נוספת היא לעדכן את ה- Channel שיצרנו בעבר ה- LED ולהוסיף לו RESOURCE חדש. כמו כן ניתן

Edit channel properties

Public

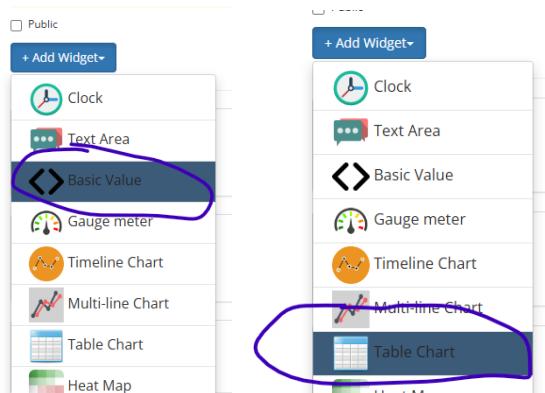
Configured resources

led		boolean	<input checked="" type="checkbox"/> SOS
sensor		number	<input checked="" type="checkbox"/> SOS
servo		boolean	<input checked="" type="checkbox"/> SOS
getdata		string	<input checked="" type="checkbox"/> SOS

+ Resource

Cancel Save

בשלב הבא נוסיף ללוח הבקרה שיצרנו כבר שני רכיבים חדשים:



נקבל לוח בקרה כולל את הרכיבים הבאים:

The screenshot shows the Beebotte ESP32 Dashboard. On the left sidebar, there are links for Channels, Dashboards (selected), Beerules (beta), Console, Account Settings, Account Usage, and Support.

The main area displays three cards:

- My LED**: Shows a switch labeled "ON" which was last updated 20 hours ago.
- esp32 servo**: Shows a switch labeled "ON" which was last updated 19 hours ago.
- esp32.getdata**: Shows a message card with the text "Hello Gadi" from 37 minutes ago. Below it is a table of data for the "esp32.getdata" resource.

The table has columns: RESOURCE, DATA, and WHEN. The data is as follows:

RESOURCE	DATA	WHEN
esp32.getdata	Hello Gadi	37 minutes ago
esp32.getdata	Hello Gadi	38 minutes ago
esp32.getdata	Hello Gadi	38 minutes ago
esp32.getdata	Hello Gadi	38 minutes ago
esp32.getdata	Ford111	17 hours ago
esp32.getdata	Ford111	17 hours ago

בשלב הבא נכתוב את הקוד הבא:

```
# File name: mqtt8.py
from machine import Timer
from umqtt.simple import MQTTClient
import utime
import os
import sys
import json

PUBLISH_PERIOD    = 5
timeSave = 0

CHANNEL_TOKEN = 'token_jr_____ku'
CHANNEL_NAME   = 'esp32'
RESOURCE_NAME  = 'getdata'
MQTT_SERVER    = 'mqtt.beebotte.com'
MQTT_USER      = 'token:' + CHANNEL_TOKEN
MQTT_TOPIC     = CHANNEL_NAME + '/' + RESOURCE_NAME

def handleTimerInt(timer):
    global timeSave
```

```

thisdict = {
    "data": "Hello Gadi",
    "write": True
}
msg= json.dumps(thisdict, separators=', ', ':')
client.publish(MQTT_TOPIC,msg, qos=0)
print("Publish:",msg, "MQTT_TOPIC=",MQTT_TOPIC)
timeSave = 0

# create a random MQTT clientID
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client'+str(random_num), 'utf-8')
client = MQTTClient(mqtt_client_id, MQTT_SERVER, user=MQTT_USER,
password='', keepalive=PUBLISH_PERIOD*2 )
myTimer = Timer(0)

try:
    client.connect()
    myTimer.init(period=PUBLISH_PERIOD*1000, mode=Timer.PERIODIC,
callback=handleTimerInt)
except Exception as e:
    print('could not connect to MQTT server {}{}'.format(type(e).__name__,
e))
    sys.exit()

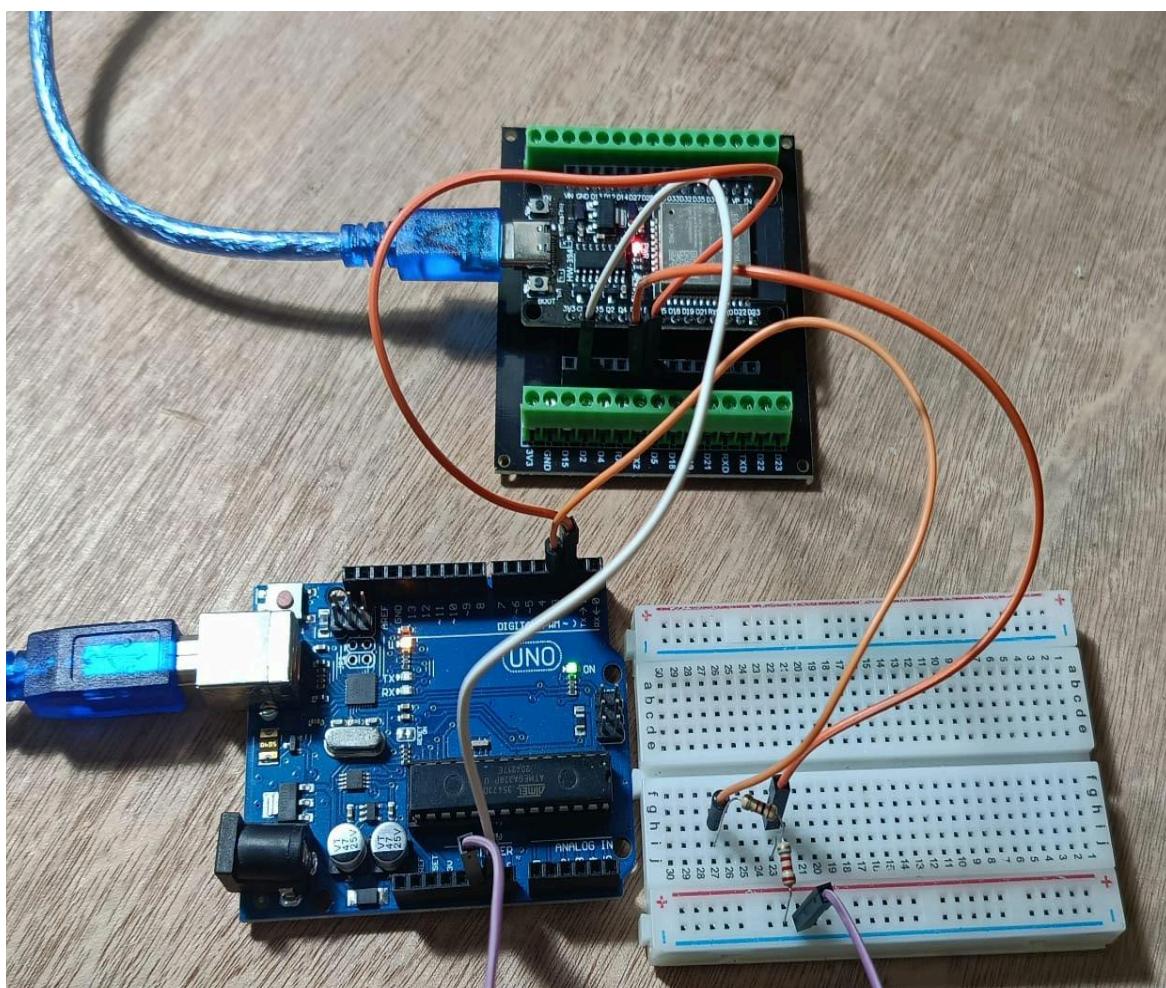
while True:
    try:
        print('Publish new data in', (PUBLISH_PERIOD-timeSave) , "second.",
", end="\r")
        utime.sleep(1)
        timeSave+=1
    except KeyboardInterrupt:
        print('Ctrl-C pressed...exiting')
        client.disconnect()
        sys.exit()

```

נקבל את הפלט הבא:

esp32.getdata		
Hello Gadi		
41 minutes ago		
esp32.getdata		
RESOURCE	DATA	WHEN
esp32.getdata	Hello Gadi	41 minutes ago
esp32.getdata	Hello Gadi	41 minutes ago
esp32.getdata	Hello Gadi	41 minutes ago
esp32.getdata	Hello Gadi	41 minutes ago
esp32.getdata	Ford111	17 hours ago

עד עכשיו רأינו כיצד ניתן להעביר מחרוזת טקסט מבקר ESP32 לענן של [.beebotte](#).
בשלב הבא נחבר בתקשורת טוירית מבוססת UART בין בקר ארדואינו לבין ESP32 כדי לקבל מבקר ארדואינו מחרוזת טקסט ולהעביר אותה לענן של [.beebotte](#) דרך ESP32 המהווה ערכף התקשורת לאינטרנט.



להלן הקוד:

```
#file name mqtt9.py
import uasyncio as asyncio
from machine import UART
from machine import Timer, UART, reset
from umqtt.simple import MQTTClient
import os
import sys
import json

uart = UART(2, 115200)
PING_PERIOD = 120

CHANNEL_TOKEN = 'token_jr_____ku'
CHANNEL_NAME = 'esp32'
RESOURCE_NAME = 'getdata'
MQTT_SERVER = 'mqtt.beebotte.com'
MQTT_USER = 'token:' + CHANNEL_TOKEN
MQTT_TOPIC = CHANNEL_NAME + '/' + RESOURCE_NAME

def handleTimerInt(timer):
    client.ping()
    print('Ping to MQTT server')

async def waiting():
    ch = ['-','\\','|','/']
    while True:
        for item in ch:
            print(item, end='\r')
            await asyncio.sleep(0.5)

async def receiver():
    print("Starts listening to serial communication")
    sreader = asyncio.StreamReader(uart)
    res =""
    while True:
        res = await sreader.readline()
        print('Recieved', res)
        myDict = {
            "data": res,
            "write": True
        }
        msg= json.dumps(myDict, separators=', ', ':')
        client.publish(MQTT_TOPIC,msg, qos=0)
        print("Publish:",msg, "MQTT_TOPIC=",MQTT_TOPIC)

# create a random MQTT clientID
random_num = int.from_bytes(os.urandom(3), 'little')
mqtt_client_id = bytes('client_'+str(random_num), 'utf-8')
client = MQTTClient(mqtt_client_id, MQTT_SERVER, user=MQTT_USER,
password='', keepalive=PING_PERIOD*2 )
myTimer = Timer(0)
```

```

try:
    client.connect()
    myTimer.init(period=PING_PERIOD*1000, mode=Timer.PERIODIC,
callback=handleTimerInt)
except Exception as e:
    print('could not connect to MQTT server {}'.format(type(e).__name__,
e))
    sys.exit()

print("Connect to MQTT server!")

try:
    loop = asyncio.get_event_loop()
    loop.create_task(waiting())
    loop.create_task(receiver())
    loop.run_forever()

except KeyboardInterrupt:
    print('Ctrl-C pressed... exiting')
    client.disconnect()
    reset()

```

בקר UNO מכיל את הקוד הבא:

```

#include <SoftwareSerial.h>

// RX is digital pin 2 (connect to TX of other device)
// TX is digital pin 3 (connect to RX of other device)
SoftwareSerial ESP32Serial(2, 3); // RX, TX

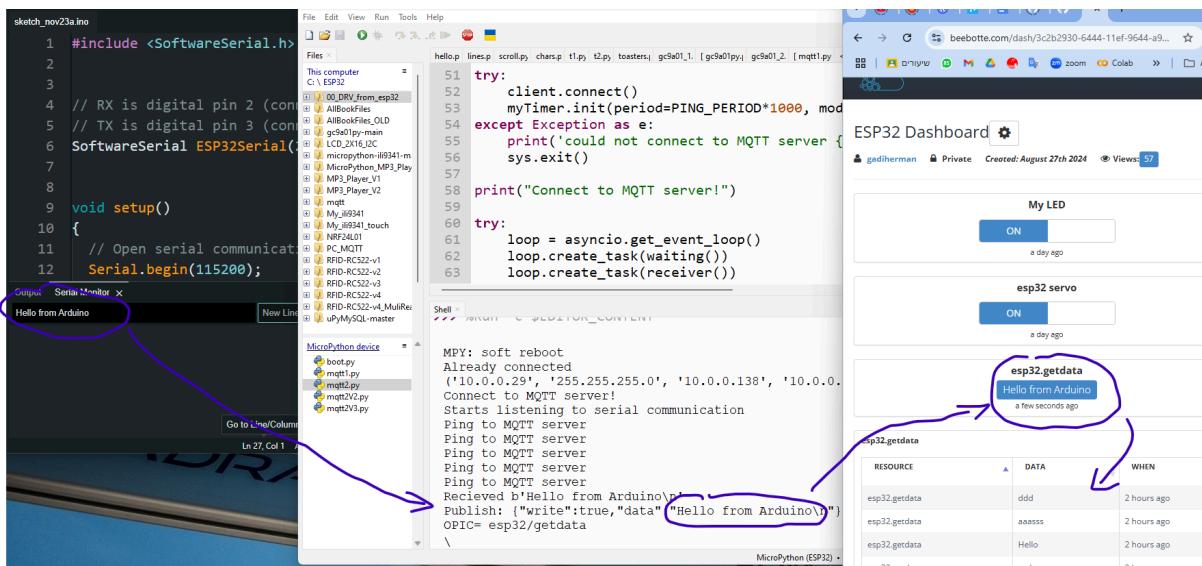
void setup()
{
    // Open serial communications and wait for port to open:
    Serial.begin(115200);

    // set the data rate for the SoftwareSerial port
    ESP32Serial.begin(115200);
}

void loop() // run over and over
{
    if (ESP32Serial.available())
        Serial.write(ESP32Serial.read());
    if (Serial.available())
        ESP32Serial.write(Serial.read());
}

```

נקבל את הפלט הבא:



משימה 17 - הפעלת רכיב השמעת קבצי MP3 מובוס על YX5300

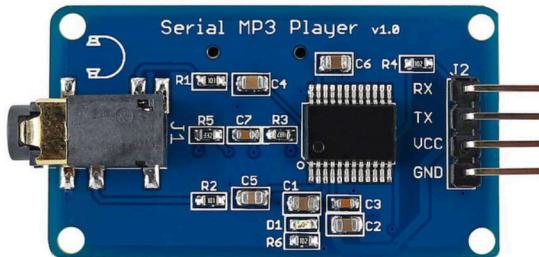
קישור:

https://github.com/GadiHerman/ESP32_YX5300_Serial_MP3_Player

נגן MP3 מובוס על רכיב YX5300 תומך בהשמעת קבצי בפורמט MP3 ו-WAV בתדר דגימה של 8kHz עד 48kHz. קבצי השמע מאוחסנים בכרטיסי מיקרו SD שמתחבר לשקע כרטיס TF בגב הלוח. הנגן כולל בקר פנימי השולט על השמעת MP3 על ידי שליחת פקודות טוריות למודול דרך יציאת UART.

להלן מאפייני הרכיב:

Item	Min	Typical	Max	Unit
Power Supply(VCC)	3.2	5	5.2	VDC
Current (@VCC=5V)	/	/	200	mA
Logic interface	3.3V / 5V TTL			/
Supported Card Type	Micro SD card(<=2G); Mirco SDHC card(<=32G)			/
File system format	Fat16 / Fat32			/
Uart baud rate	9600			bps



הרכיב כולל מספר הדקים:

1. ממוקם הבקרה הכלול את ההדים GND, VCC, TX, RX. הדק GND לאדמה של ה- ESP32 הדק VCC מחובר ל- 5V. הדק TX (שידור) מחובר להדק RX של בקר RX (קבלה) מחובר להדק TX של בקר ESP32 .
2. שקע כרטיס TF בצד האחורית של ה-PCB לחיבור כרטיס מיקרו SD הכלול קבצי MP3/WAV.
3. מחוון השמעה (LED יירוק) מהבהבת בזמן ההשמעה. דלוק קבוע בכל זמן אחר.
4. שקע פלט קול לאוזניות או מגבר חיצוני.

לבקר ESP32 יש 3 ממוקמי תקשורת UART (אחד זמן לשימוש ישיר) על פי המפרט הבא:

UART0: (GPIO 1 and GPIO3) משמש לתקשורת מול המחשב -

UART1: (GPIO 9 and GPIO10) – connected to the ESP32 SPI flash memory, so you can't use them.

זמן לשימוש - UART2: (GPIO 17 and GPIO 16)

כמו כן ניתן להשתמש ב-UART1 אך נדרש לספק לו את מספרי הבדיקה של tx ו-rx בעת ייצירת עצם חדש של המחלקה UART באופן הבא:

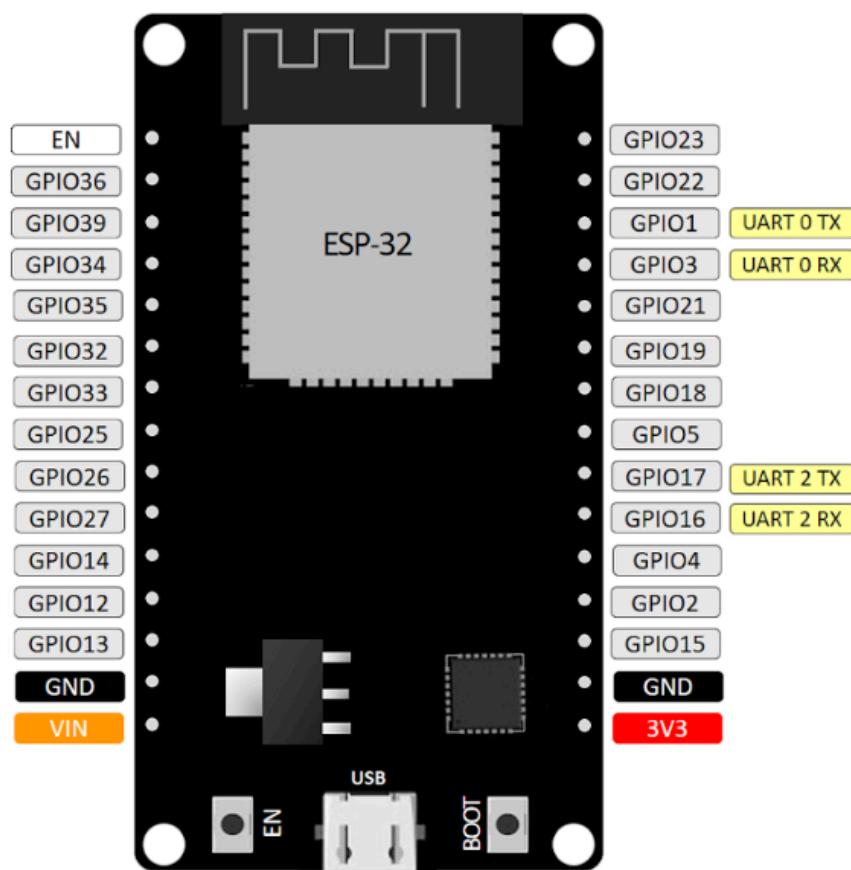
```
uart = UART(1, baudrate=9600, tx=19, rx=18)
```

ניתן לראות שלמרות שהדק בירית המוחדר של UART1 הוא tx=10 ו-rx=9 (התפקידים !!!)

החלפנו אותם להדקים tx=19, rx=18.

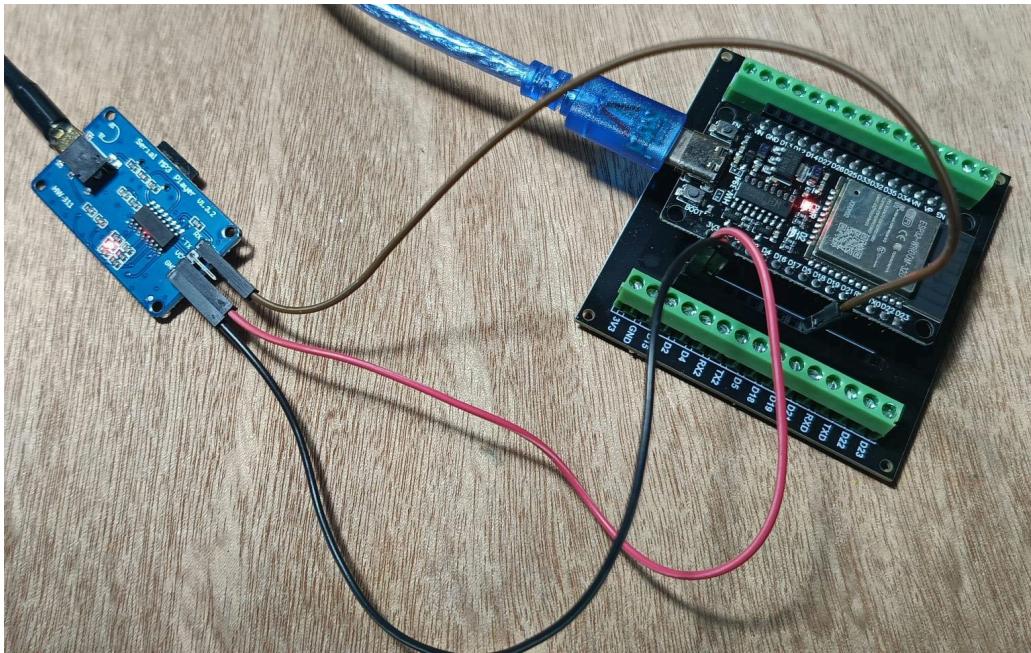
	UART0	UART1	UART2
tx	1	10	17
rx	3	9	16

להלן מיקום הדק ה-UART בברker:



כליורו UART0 לא זמין, UART2 זמין דרך הבדיקה tx=17 ו-rx=16 ו-UART1 זמין לשימוש אך דורש קביעת הבדיקה שונה מבירית המוחדר.

להלן דוגמה לאופן החיבור:

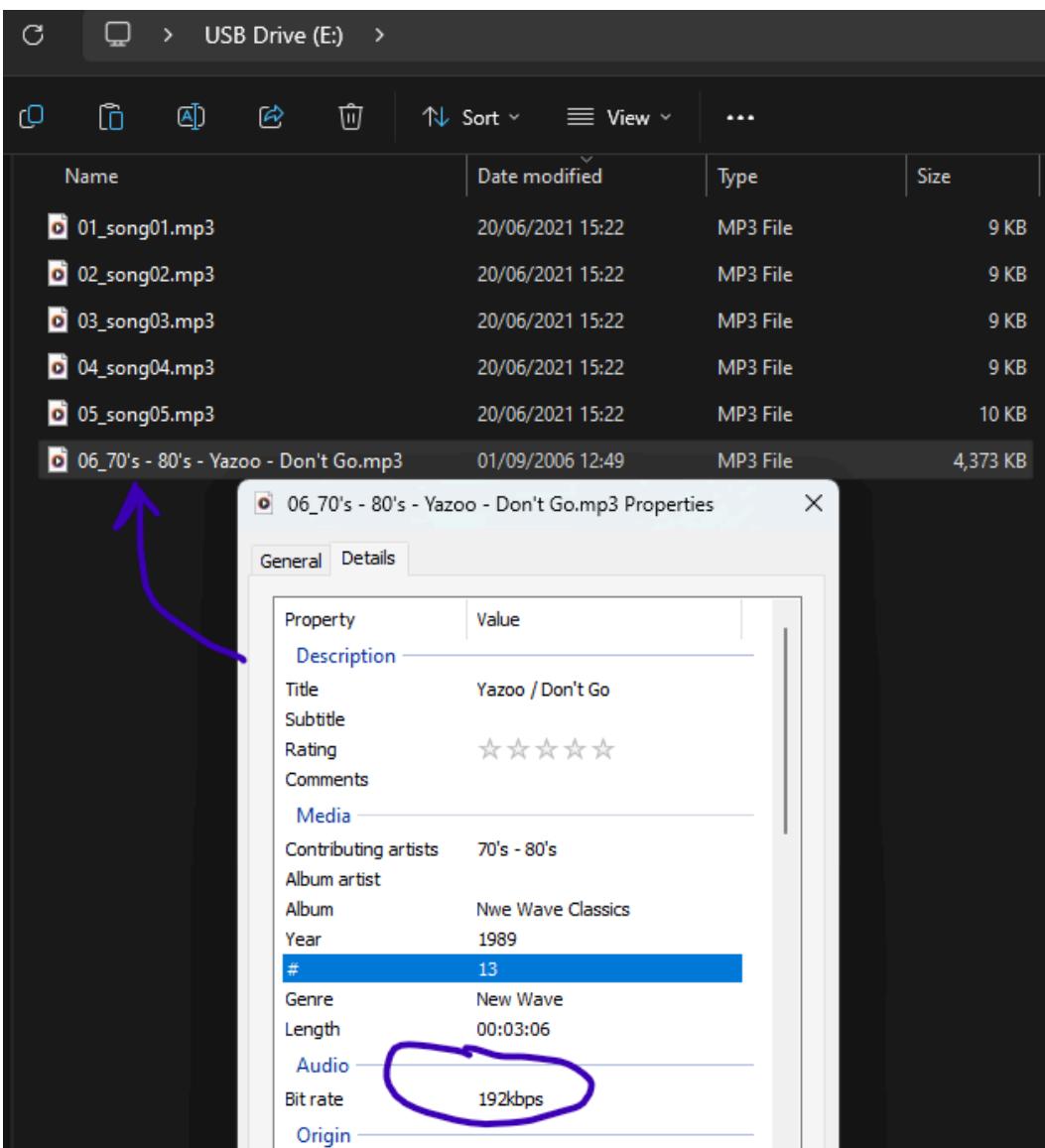


שים לב!!! יש צורך לחבר את הדק AD של הבקר להדק RX של רכיב ה- MP3

יש לארגן את קבצי השמע בזיכרון ה- SD על פי הפורמט הבא:

```
+-- 01
|   + 001-file_description.mp3
|   + 002-file_description.mp3
|
+- 02
|   + 003-file_description.mp3
|   + 004-file_description.mp3
|   + 005-file_description.mp3
|
+- 03
    + 006-file_description.mp3
    + 007-file_description.mp3
```

שים לב! לצורך הפשטות ניתן להעתיק את כל הקבצים לספריית השורש (כלומר ללא שימוש בתיקיות) באופן הבא:



להלן מימוש מחלקת המטפלת בתקשרות על הרכיב. יש ליצור קובץ בשם MD_YX5300.py. MD_YX5300 הכלול בתוכן הבא:

```
from machine import UART
from time import sleep

class MD_YX5300:
    def __init__(self, Uart=UART(2, 9600)):
        self.command=bytearray()
        self.command.append(0x7e)
        self.command.append(0xFF)
        self.command.append(0x06)
        self.command.append(0x00)
        self.command.append(0x00)
        self.command.append(0x00)
        self.command.append(0xEF)

        self uart = Uart
        #self.uart = UART_NUMBER, 9600
```

```

# set volume to mid point (0=min 30=max)
self.volume_level=30
self.set_volume(self.volume_level)
sleep(0.5)

def play_next(self):
    self.command[3]=0x01
    self.uart.write(self.command)

def play_previous(self):
    self.command[3]=0x02
    self.uart.write(self.command)

def play_track(self,track_id):
    self.command[3]=0x03
    self.command[6]=track_id
    self.uart.write(self.command)

def play(self):
    self.command[3]=0x03
    self.command[6]=1
    self.uart.write(self.command)

def volume_up(self,step_count=1):
    if self.volume_level<=(30-step_count):
        self.volume_level=self.volume_level+step_count
    else:
        self.volume_level=30
    self.set_volume(self.volume_level)

def volume_down(self,step_count=1):
    if self.volume_level>=step_count:
        self.volume_level=self.volume_level-step_count
    else:
        self.volume_level=0
    self.set_volume(self.volume_level)

def set_volume(self,level):
    self.command[3]=0x06
    self.command[6]=level
    self.uart.write(self.command)

def sleep_module(self):
    self.command[3]=0x0A
    self.uart.write(self.command)

def wakeup_module(self):
    self.command[3]=0x0B
    self.uart.write(self.command)

def reset_module(self):
    self.command[3]=0x0C
    self.uart.write(self.command)

```

```

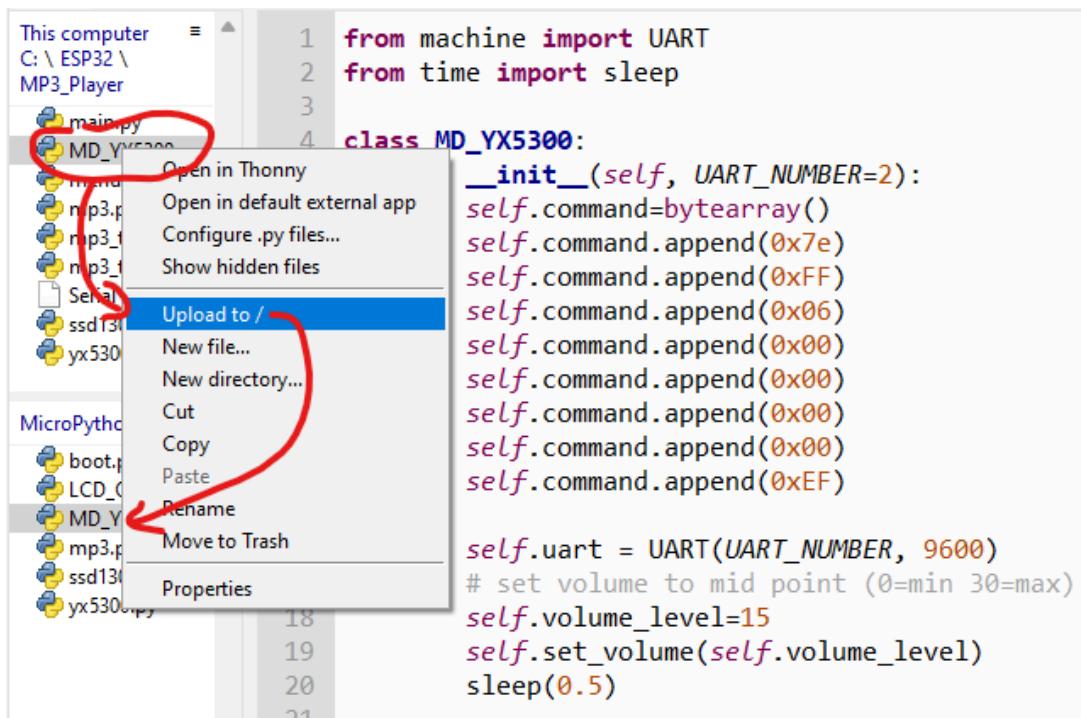
def pause(self):
    self.command[3]=0x0E
    self.uart.write(self.command)

def resume(self):
    self.command[3]=0x0D
    self.uart.write(self.command)

def stop(self):
    self.command[3]=0x16
    self.uart.write(self.command)

```

כדי לבדוק את הרכיב יש להעתיק את קובץ המחלקה לבקר על ידי ביצוע השלבים הבאים:



להלן קוד בדיקה הבסיסי לרכיב (העובד דרך UART2 כבריתת מחדל):

```

import MD_YX5300

mp3 = MD_YX5300.MD_YX5300()
mp3.play_track(1)

```

כדי להחליף לתקשרות עם UART1 יש להיעזר בקוד הבא:

```

from machine import UART
import MD_YX5300

uart = UART(1, baudrate=9600, tx=19, rx=18)
#uart = UART(2, 9600)
mp3 = MD_YX5300.MD_YX5300(uart)

```

```
mp3.play_track(1)
```

להלן דוגמת קוד למימוש נגן הcolel קולט ופלט דרך חלון ה- Shell

```
import MD_YX5300

def display_menu(menu_items):
    print("==== MD_YX5300 Test Menu ====")
    for i, item in enumerate(menu_items, 1):
        print(f"{i}. {item}")
    print("===== ")

def get_user_choice(menu_items):
    while True:
        try:
            choice = int(input("Enter the number of your choice (0 to exit): "))
            if choice == 0:
                return None
            if 1 <= choice <= len(menu_items):
                return menu_items[choice - 1]
            else:
                print("Invalid choice. Please try again.")
        except ValueError:
            print("Invalid input. Please enter a number.")

def main():
    menu_items = [
        "Play Next Track",
        "Play Previous Track",
        "Play Track by ID",
        "Play First Track",
        "Volume Up",
        "Volume Down",
        "Set Volume",
        "Sleep Module",
        "Wakeup Module",
        "Reset Module",
        "Pause",
        "Resume",
        "Stop"
    ]
    player = MD_YX5300.MD_YX5300(UART_NUMBER=2)

    while True:
        display_menu(menu_items)
        choice = get_user_choice(menu_items)

        if choice is None:
            print("Exiting the test menu. Goodbye!")
            break
```

```

if choice == "Play Next Track":
    player.play_next()
elif choice == "Play Previous Track":
    player.play_previous()
elif choice == "Play Track by ID":
    track_id = int(input("Enter the track ID to play: "))
    player.play_track(track_id)
elif choice == "Play First Track":
    player.play()
elif choice == "Volume Up":
    player.volume_up(2)
elif choice == "Volume Down":
    player.volume_down(2)
elif choice == "Set Volume":
    volume_level = int(input("Enter the volume level (0-30): "))
    player.set_volume(volume_level)
elif choice == "Sleep Module":
    player.sleep_module()
elif choice == "Wakeup Module":
    player.wakeup_module()
elif choice == "Reset Module":
    player.reset_module()
elif choice == "Pause":
    player.pause()
elif choice == "Resume":
    player.resume()
elif choice == "Stop":
    player.stop()

if __name__ == "__main__":
    main()

```

נקבל את הפלט הבא:

```

MicroPython dev1 = 
boot.py
LCD_OLED.py
MD_YX5300.py
mp3.py
ssd1306.py
yx5300.py

44
45     if choice is None:
46         print("Exiting the test menu. Goodbye!")
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Already connected
('10.0.0.20', '255.255.255.0', '10.0.0.138', '10.0.0.138')
==== MD_YX5300 Test Menu ====
1. Play Next Track
2. Play Previous Track
3. Play Track by ID
4. Play First Track
5. Volume Up
6. Volume Down
7. Set Volume
8. Sleep Module
9. Wakeup Module
10. Reset Module
11. Pause
12. Resume
13. Stop
=====
Enter the number of your choice (0 to exit):

```

משימה 18 - הפעלת צג גרפי 2x16 I2C LCD

במשימה זה תלמד כיצד להשתמש בתצוגת LCD 2x16 עם בקר ESP32 כדי להציג טקסט ומספרים כתווים על המסך. הצג בגודל 16x2 תווים יכול להציג 16 תווים בשורה על פני שתי שורות של תווים. LCD-הברker מסוג 008044780hd, כמו כן יש לו גם מודול I2C שמחובר אליו ובכך מקל על החיבור בין הצג לברker ESP32.

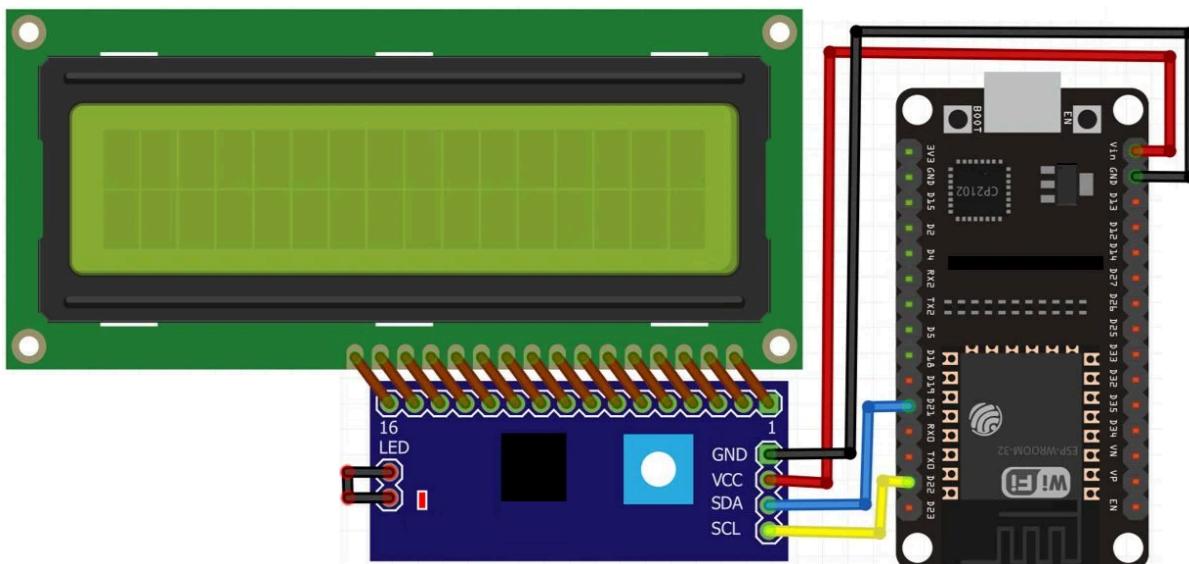
קישורים:

<https://microcontrollerslab.com/i2c-lcd-esp32-esp8266-micropython-tutorial/>

חיבור הצג לברker

OLED	ESP32
Vin	3.3V
GND	GND
SCL	GPIO 22
SDA	GPIO 21

شرطוט חשמלי של החיבור:



בשלב הראשון נעזר בקוד הבא כדי לאתר את כתובות הרכיב:

```
from machine import I2C, Pin  
import machine
```

```

i2c = I2C(scl=Pin(22), sda=Pin(21))

devices = i2c.scan()

if len(devices) == 0:
    print("No i2c device !")
else:
    print('i2c devices found:',len(devices))
for device in devices:
    print("At address: ",hex(device))

```

נקבל את הפלט הבא:

```

>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Already connected
('10.0.0.20', '255.255.255.0', '10.0.0.138', '10.0.0.138')
Warning: I2C(-1,...) is deprecated, use SoftI2C(...) instead
i2c devices found: 2
At address: 0x27
At address: 0x3c
>>>

```

ניתן לראות שיש לנו 2 רכיבים המתחברים על אותו ערוץ I2C האחד בכתובת 27 והשני 3CH.

תוכן הקובץ עטוף :

```

from utime import sleep_ms
from machine import I2C

class I2C_LCD():
    def __init__(self, i2c, addr = 0x27):
        self.i2c=i2c
        self.buf = bytearray(1)
        self.BK = 0x08
        self.RS = 0x00
        self.E = 0x04
        self.ADDR = addr
        self.setcmd(0x33)
        sleep_ms(5)
        self.send(0x30)
        sleep_ms(5)
        self.send(0x20)
        sleep_ms(5)
        for i in [0x28, 0x0C, 0x06, 0x01]:
            self.setcmd(i)
        self.px, self.py = 0, 0

```

```

        self.pb = bytearray(16)

    def setReg(self, dat):
        self.buf[0] = dat
        self.i2c.writeto(self.ADDR, self.buf)
        sleep_ms(1)

    def send(self, dat):
        d=(dat&0xF0)|self.BK|self.RS
        self.setReg(d)
        self.setReg(d|0x04)
        self.setReg(d)

    def setcmd(self, cmd):
        self.RS=0
        self.send(cmd)
        self.send(cmd<<4)

    def setdat(self, dat):
        self.RS=1
        self.send(dat)
        self.send(dat<<4)

    def autoaddr(self):
        for i in range(32, 63):
            try:
                if self.i2c.readfrom(i, 1):
                    return i
            except:
                pass
        raise Exception('I2C address detect error!')

    def write_cgram(self, buf, reg=0):
        n = len(buf)
        self.setcmd(0x40 + (reg%8)*8)
        for i in range(n):
            self.setdat(buf[i])

    def clear(self):
        self.setcmd(1)

    def backlight(self, on):
        if on:
            self.BK=0x08
        else:
            self.BK=0
        self.setcmd(0)

    def on(self):
        self.setcmd(0x0C)

    def off(self):
        self.setcmd(0x08)

```

```

def shl(self):
    self.setcmd(0x18)

def shr(self):
    self.setcmd(0x1C)

def char(self, ch, x=-1, y=0):
    if x>=0:
        a=0x80
        if y>0:
            a=0xC0
        self.setcmd(a+x)
    self.setdat(ch)

def puts(self, s, x=0, y=0):
    if type(s) is not str:
        s = str(s)
    if len(s)>0:
        self.char(ord(s[0]),x,y)
        for i in range(1, len(s)):
            self.char(ord(s[i]))

def newline(self):
    self.px = 0
    if self.py < 1:
        self.py += 1
    else:
        for i in range(16):
            self.char(self.pb[i], i)
            self.char(32, i, 1)
            self.pb[i] = 32
def print(self, s):
    if type(s) is not str:
        s = str(s)
    for i in range(len(s)):
        d = ord(s[i])
        if d == ord('\n'):
            self.newline()
        else:
            self.char(d, self.px, self.py)
            if self.py:
                self.pb[self.px] = d
            self.px += 1
            if self.px > 15:
                self.newline()

```

תוכן הקובץ HelloWorld.py

```
from machine import I2C, Pin
from i2c_lcd import I2C_LCD
from time import sleep_ms

i2c = I2C(scl=Pin(22), sda=Pin(21))
LCD = I2C_LCD(i2c,0x27)

LCD.print("ABCDEFGHIJKLM NOPQRSTUVWXYZ")
sleep_ms(4000)
LCD.clear()
LCD.puts("Hello Word (-: ")
n = 0
while 1:
    LCD.puts(n, 0, 1)
    n += 1
    sleep_ms(1000)
```

להלן הפלט על הציג:



משימה 19 - הפעלת צג LCD גרפי צבעוני 240*320 פיקסלים מבוסס על ILI9341

במשימה זו נממש קוד להפעלת צג גרפי צבעוני בגודל 240*320 פיקסלים מבוסס על ILI9341. הצג מחובר לבקר דרך ממשק SPI.

קישורים:

<https://github.com/rdagger/micropython-ili9341>



תקשורת SPI בbbc ESP32

תקשורת SPI - Serial Peripheral Interface היא פרוטוקול תקשורת טורי סינכרוני המשמש לתקשורת בין מיקרו-בקרים וחתקנים היקפיים. ESP32 תומך בעד 4 ערוצי SPI שונים (SPI0-SPI3).

קווי התקשורת העיקריים:

- SCLK - אות שעון
- MOSI - Master Out Slave In
- MISO - Master In Slave Out
- CS/SS - בחרית החתון

הדקם ב-2: ESP32 מציע מספר אפשרויות לחברו SPI:

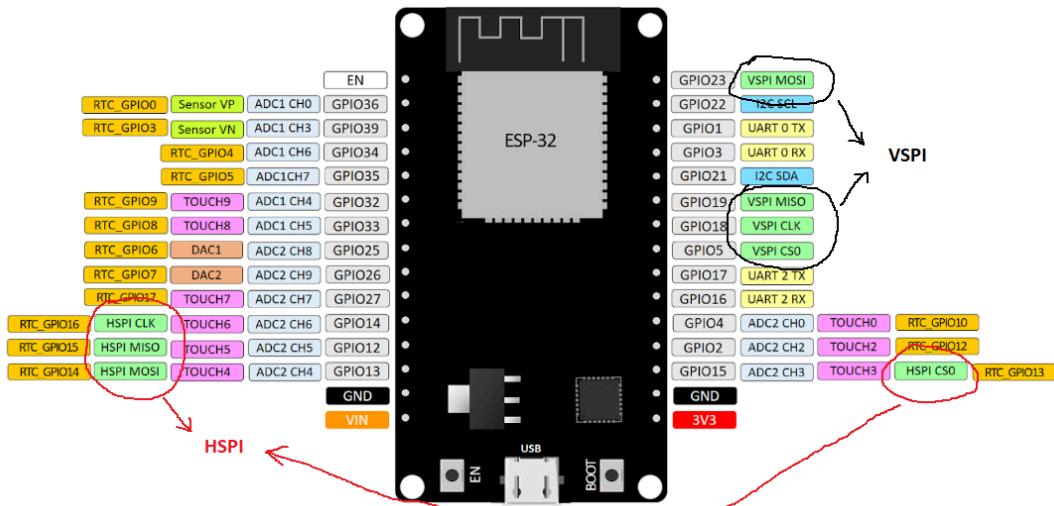
.1 SPI3 - VSPI - ברירת המחדל:

- SCLK: GPIO18
- MOSI: GPIO23
- MISO: GPIO19
- CS: GPIO5

:HSPI - SPI2 .2

- SCLK: GPIO14
- MOSI: GPIO13
- MISO: GPIO12
- CS: GPIO15

מייפוי רגילים:



שימושים נפוצים:

- תקשורת עם צגים
- קריאה/כתיבה לכרטיס SD
- תקשורת עם חיישנים
- תקשורת עם מודול RF
- העברת נתונים להתקנים היקפיים

להלן דוגמת קוד לשימוש בתקשרות SPI:

```
from machine import Pin, SPI
import time

class SPIExample:
    def __init__(self):
        # הגדלת פינוי SPI
        self.sck = Pin(18, Pin.OUT)      # שעון #
        self.mosi = Pin(23, Pin.OUT)     # מוצא נתונים #
        self.miso = Pin(19, Pin.IN)      # כניסה נתונים #
        self.cs = Pin(5, Pin.OUT)        # בחירת רכיב #

    # אתחול אובייקט SPI
    self.spi = SPI(2, baudrate=1000000,
                  polarity=0, phase=0,
                  sck=self.sck,
                  mosi=self.mosi,
                  miso=self.miso)

    def write_data(self, data):
        """ SPI פונקציה לשילוח נתונים דרך CS הפעלה # (active low)
        """
        self.cs.value(0) # CS הפעלה #
        self.spi.write(bytes(data))
        self.cs.value(1) # ביטול CS
```

```

def read_data(self, bytes_count):
    """
    פונקציה לקרוא נתונים דרך SPI
    """
    self.cs.value(0)
    data = self.spi.read(bytes_count)
    self.cs.value(1)
    return data

def transfer_data(self, data):
    """
    פונקציה לשילוח וקבלת בו-זמןית
    """
    self.cs.value(0)
    received = self.spi.write_readinto(bytes(data), bytes(len(data)))
    self.cs.value(1)
    return received

# דוגמת שימוש
def main():
    spi_example = SPIExample()

    # דוגמה לשילוח נתונים #
    data_to_send = [0x55, 0xAA, 0x0F]
    print("שלח נתונים:", data_to_send)
    spi_example.write_data(data_to_send)

    # דוגמה לקריאה נתונים #
    received_data = spi_example.read_data(3)
    print("התקבלו נתונים:", list(received_data))

    # דוגמה להעברה דו-כיוונית #
    transfer_data = [0x12, 0x34, 0x56]
    received = spi_example.transfer_data(transfer_data)
    print("נתונים שהועברו:", transfer_data)
    print("נתונים שהתקבלו:", list(received))

if __name__ == "__main__":
    main()

```

הסברים נוספים על הקוד:

1. אתחול SPI:

- בחירת עroz
- הגדרת קצב השעון (baudrate)
- קביעת הקוטביות והפאה
- הגדרת הפינים

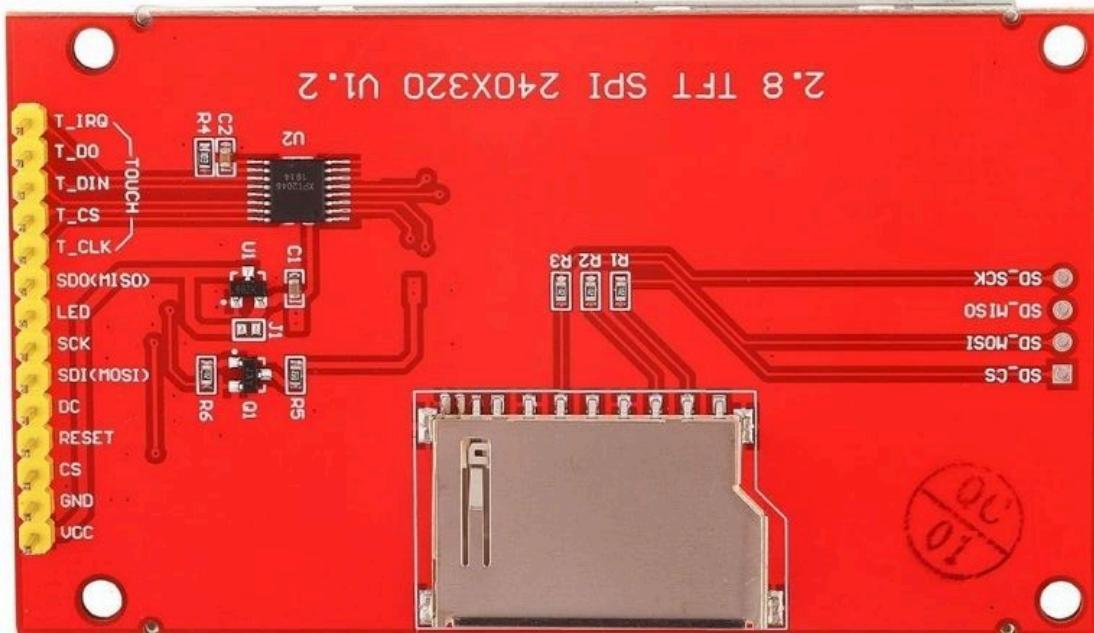
2. פעולות בסיסיות:

- write - שליחת נתונים
- read - קריאת נתונים
- write_readinto - העברת דו-כיוונית

3. טיפים לשימוש:

- תמיד לשחרר את ה-CS בסיום התקשרות
- להתאים את מהירות השעון להתקן המחבר
- לוודא חיבור נכון של הפינים

להלן מערכ החיבורים בין הציג לבקר:



ESP32	הסבר הבדיקה	ILI9341 LCD color display
D2	הדק פסיקה של מסך מגע	T_IRQ
D19 (MISO)	הדק MISO של מסך מגע	T_DO
D5	הדק בחירת רכיב של מסך מגע	T_CS
D18 (SCK)	הדק השעון של מסך מגע	T_CLK
D23 (MOSI)	הדק MOSI של מסך מגע	SDO(MISO)
5V	מתוך מוקור חיצוני של 5V	LED
D4	הדק השעון של הציג	SCK
D13	הדק MOSI של הציג	SDI(MOSI)
D4		D/C
D27	הדק איפואו הציג	RESET
D15	הדק בחירת רכיב של המסך	CS
GND	הדק GND	GND
3.3-5V	מתוך מוקור חיצוני של 3.3-5V	VCC

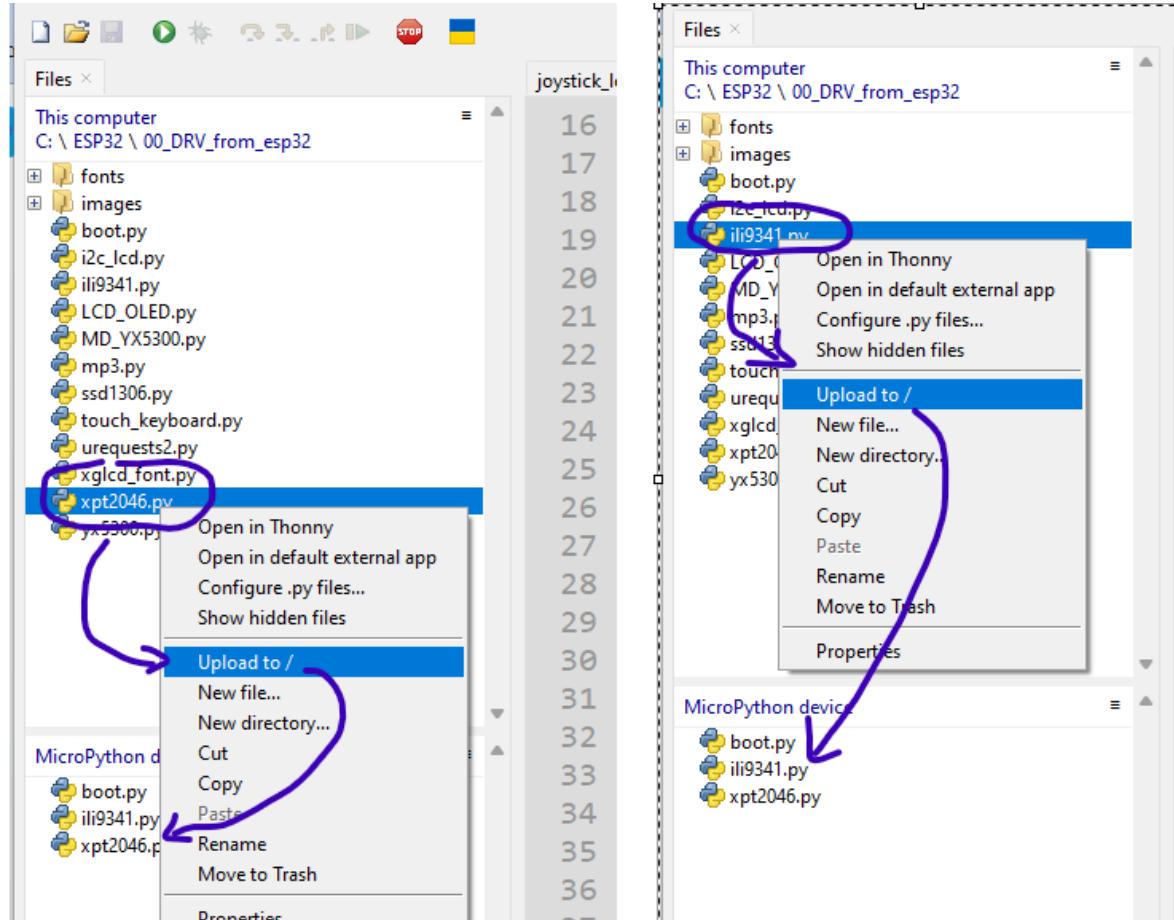
כדי להפעיל את המסר יש להוריד את הקובץ `ili9341.py` מכתובת האינטרנט הבא:

https://github.com/GadiHerman/ESP32_MicroPython_AllBookFiles/blob/main/19_LCD_ILI9341/ili9341.py

כדי להפעיל את מסך המגע יש להוריד את הקובץ `xpt2046.py` מכתובת האינטרנט הבא:

https://github.com/GadiHerman/ESP32_MicroPython_AllBookFiles/blob/main/19_LCD_ILI9341/xpt2046.py

את הקבצים `py` ו- `.xpt2046.py` יש להעביר לבקר על ידי ביצוע הפעולות הבאות:



הערה: ניתן להפעיל את המסר ללא רכיב המגע שלו, במצב זה אין צורך לחבר את הדקן המגע של המסר (אלה המסתומנים בכחול בטבלת החיבורים בין הציג לבקר) ואין צורך להוריד את הקובץ `xpt2046.py`.

דוגמאות קוד להפעלת הצג

כתבת טקסט על המסך:

```
from time import sleep
from ili9341 import Display, color565
from machine import Pin, SPI, reset

spi = SPI(1, baudrate=40000000, sck=Pin(14), mosi=Pin(13))
display = Display(spi, dc=Pin(4), cs=Pin(15), rst=Pin(27))

display.draw_text8x8(0, 0, 'Hello from', color565(255, 0, 255))
display.draw_text8x8(16, 16, 'ESP32', color565(255, 255, 0))
```

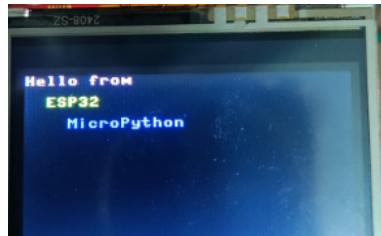
```

display.draw_text8x8(32, 32, 'MicroPython', color565(0, 0, 255))

sleep(15)
display.cleanup()
reset()

```

להלן הפלט:



נדגים תכונות נוספות להציג טקסט כמו צבע רקע וויבוב הטקסט על המסך.

```

from time import sleep
from ili9341 import Display, color565
from machine import Pin, SPI, reset

spi = SPI(1, baudrate=40000000, sck=Pin(14), mosi=Pin(13))
display = Display(spi, dc=Pin(4), cs=Pin(15), rst=Pin(27))

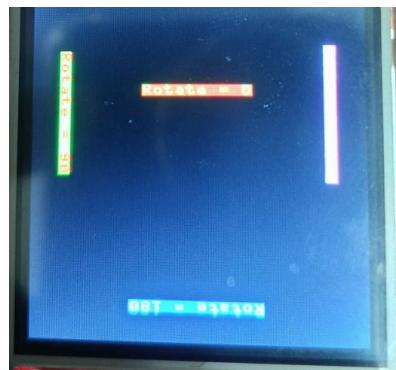
x_center = display.width // 2
y_center = display.height // 2

display.draw_text8x8(x_center - 40, 140, "Rotate = 0",
                     color565(0, 255, 0), background=color565(255, 0, 0))
display.draw_text8x8(20, y_center - 44, "Rotate = 90", color565(255, 0, 0),
                     rotate=90, background=color565(0, 255, 0))
display.draw_text8x8(x_center - 48, display.height - 29, "Rotate = 180",
                     color565(0, 255, 255), rotate=180,
                     background=color565(0, 0, 255))
display.draw_text8x8(display.width - 29, y_center - 48, "Rotate = 270",
                     color565(255, 255, 255), rotate=270,
                     background=color565(255, 0, 255))

sleep(15)
display.cleanup()
reset()

```

להלן הפלט:



מחולל צורות:

```
from time import sleep
from ili9341 import Display, color565
from machine import Pin, SPI, reset

spi = SPI(1, baudrate=40000000, sck=Pin(14), mosi=Pin(13))
display = Display(spi, dc=Pin(4), cs=Pin(15), rst=Pin(27))

display.clear(color565(64, 0, 255))
sleep(1)
display.clear()
display.draw_hline(10, 319, 229, color565(255, 0, 255))
sleep(1)
display.draw_vline(10, 0, 319, color565(0, 255, 255))
sleep(1)
display.fill_hrect(23, 50, 30, 75, color565(255, 255, 255))
sleep(1)
display.draw_hline(0, 0, 222, color565(255, 0, 0))
sleep(1)
display.draw_line(127, 0, 64, 127, color565(255, 255, 0))
sleep(2)
display.clear()
coords = [[0, 63], [78, 80], [122, 92], [50, 50], [78, 15], [0, 63]]
display.draw_lines(coords, color565(0, 255, 255))
sleep(1)
display.clear()
display.fill_polygon(7, 120, 120, 100, color565(0, 255, 0))
sleep(1)
display.fill_rectangle(0, 0, 15, 227, color565(255, 0, 0))
sleep(1)
display.clear()
display.fill_rectangle(0, 0, 163, 163, color565(128, 128, 255))
sleep(1)
display.draw_rectangle(0, 64, 163, 163, color565(255, 0, 255))
sleep(1)
display.fill_rectangle(64, 0, 163, 163, color565(128, 0, 255))
sleep(1)
display.draw_polygon(3, 120, 286, 30, color565(0, 64, 255), rotate=15)
sleep(3)
display.clear()
display.fill_circle(132, 132, 70, color565(0, 255, 0))
```

```

sleep(1)
display.draw_circle(132, 96, 70, color565(0, 0, 255))
sleep(1)
display.fill_ellipse(96, 96, 30, 16, color565(255, 0, 0))
sleep(1)
display.draw_ellipse(96, 256, 16, 30, color565(255, 255, 0))
sleep(5)
display.cleanup()
reset()

```

שימוש במסך מגע

להלן דוגמת קוד המציג על המסך הודעה הכללת את המיקום שבו אנו נוגעים במסך.

```

from ili9341 import Display, color565
from xpt2046 import Touch
from machine import idle, Pin, SPI, reset
WHITE = color565(255, 255, 255)

def touchscreen_press(x, y):
    y = (display.height - 1) - y
    display.draw_text8x8(60, 50, "X=%3d , Y=%3d" %(x, y), WHITE)
    print("X="+str(x)+" Y="+str(y))

spi1 = SPI(1, baudrate=4000000, sck=Pin(14), mosi=Pin(13))
display = Display(spi1, dc=Pin(4), cs=Pin(15), rst=Pin(27))
spi2 = SPI(2, baudrate=500000, sck=Pin(18), mosi=Pin(23), miso=Pin(19))
touch = Touch(spi2, cs=Pin(5), int_pin=Pin(2),
int_handler=touchscreen_press)

display.clear()
display.draw_text8x8(50, 70, "Touch the screen!", WHITE)

try:
    while True:
        idle()
except KeyboardInterrupt:
    print("\nCtrl-C pressed. Cleaning up and exiting...")
finally:
    display.cleanup()
    reset()

```

נקבל את הפלט הבא:



להלן דוגמת קוד להפעלת נורת LED תוך כדי לחימה על שני כפתורים המצוירים על המסך.

```

from ili9341 import Display, color565
from xpt2046 import Touch
from machine import idle, Pin, SPI, reset

# Colors
CYAN = color565(0, 255, 255)
PURPLE = color565(255, 0, 255)
WHITE = color565(255, 255, 255)
GREEN = color565(0, 255, 0)
RED = color565(255, 0, 0)

# Button dimensions
BUTTON_WIDTH = 100
BUTTON_HEIGHT = 50
BUTTON_SPACING = 20

# Initialize LED
led = Pin(22, Pin.OUT) # Adjust the pin number as needed

def draw_buttons():
    # Draw ON button
    display.fill_rectangle(10, 10, BUTTON_WIDTH, BUTTON_HEIGHT, GREEN)
    display.draw_text8x8(60 - 8, 35 - 4, "ON", WHITE)

    # Draw OFF button
    display.fill_rectangle(10 + BUTTON_WIDTH + BUTTON_SPACING, 10,
    BUTTON_WIDTH, BUTTON_HEIGHT, RED)
    display.draw_text8x8(60 + BUTTON_WIDTH + BUTTON_SPACING - 12, 35 - 4,
    "OFF", WHITE)

def touchscreen_press(x, y):

```

```

y = (display.height - 1) - y
display.draw_text8x8(display.width // 2 - 32,
                     display.height - 9,
                     "{0:03d}, {1:03d}".format(x, y),
                     CYAN)
if x>=10 and x<=10 + BUTTON_WIDTH and y>=10 and y<= 10 + BUTTON_HEIGHT:
    led.on()
    display.draw_text8x8(10, 100, "LED ON ", color565(255, 255, 255))
elif x> 10 + BUTTON_WIDTH and x< display.width - 10 and y>=10 and y<=
10 + BUTTON_HEIGHT:
    led.off()
    display.draw_text8x8(10, 100, "LED OFF ", color565(255, 255, 255))

spi1 = SPI(1, baudrate=4000000, sck=Pin(14), mosi=Pin(13))
display = Display(spi1, dc=Pin(4), cs=Pin(15), rst=Pin(27))

spi2 = SPI(2, baudrate=500000, sck=Pin(18), mosi=Pin(23), miso=Pin(19))
touch = Touch(spi2, cs=Pin(5), int_pin=Pin(2),
int_handler=touchscreen_press)

# Clear the screen
display.clear()

# Draw initial buttons
draw_buttons()

# Display initial message
display.draw_text8x8(60, 70, "Touch a button", WHITE)

try:
    while True:
        idle()
except KeyboardInterrupt:
    print("\nCtrl-C pressed. Cleaning up and exiting...")
finally:
    display.cleanup()
    reset()

```

נקבל את הפלט הבא:



משימה 20 - קרייה וכתיבה של תג RFID תוך שימוש ב- RC522

קישורים:

<https://github.com/rdagger/micropython-ili9341>

RC522 הוא קורא/כותב RFID נפוץ מאוד המיועד לקרייה וכתיבה של תג RFID וקרטיסים חכמים הפעלים בתדר של 13.56MHz. הוא מבוסס על השבב MFRC522 של חברת NXP.

מאפיינים עיקריים:

- מתח הפעלה: 3.7V
- ממשך תקשורת: SPI
- טווח קרייה: עד 5 ס"מ (תלוי בתג)
- תמיכה בתקנים: ISO 14443A/MIFARE
- צירכיט זרם נמוכה: C-A-m13 בזמן פעולה
- מידות קטנות: BD"כ 60x40 מ"מ
- מחיר נמוך יחסית

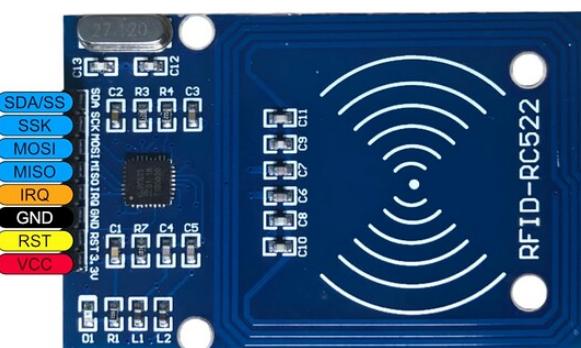
שימושים עיקריים:

1. בקרת כניסה:
 - מערכות געילה אלקטרוניות
 - זיהוי עובדים
 - בקרת גישה למתקנים
2. מערכות תשלום:
 - קרטיסי תחבורה ציבורית
 - מערכות תשלום במזנונים
 - קרטיסי חבר במועדונים

יתרונות:

- קל לשימוש ותוכנות
- תמיכה נרחבת ו眾多 זמינות
- אמינות גבוהה
- מחיר משתלם

חיבור בסיסי: הרכיב מתחבר למקroit-בקר דרך ממשך SPI. להלן הדקן הרכיב:



dagshim leshimush:

1. יש להקפיד על חיבור למתח 3.3V בלבד
2. מומלץ להשתמש בספירות מוכנות
3. יש לשמר על מרחק מינימלי בין קוראים במדיה ומשתמשים ביוטר אחד
4. רצוי להוסיף קבל סינון על קו המתח

תקשורת SPI בבקר ESP32

תקשורת SPI - Serial Peripheral Interface היא פרוטוקול תקשורת טורי סינכרוני המשמש לתקשורת בין מיקרו-בקרים והתקנים היקפיים. ESP32 תומך עד 4 ערוצי SPI שונים (SPI0-SPI3).

קווי התקשורת העיקריים:

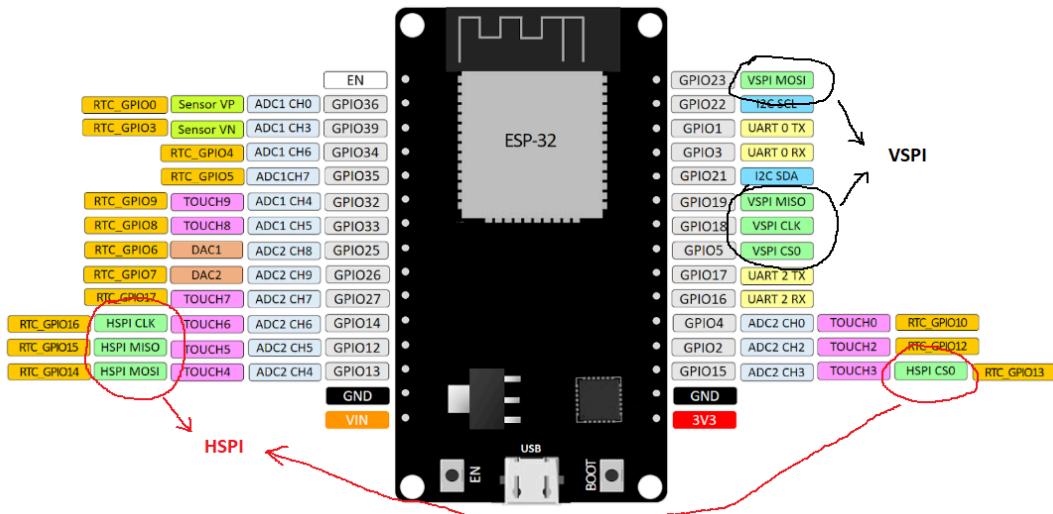
- SCLK - אות שעון
- MOSI - Master Out Slave In
- MISO - Master In Slave Out
- CS/SS - בחרית ההתקן

הדקם ב-ESP32: ESP32 מציע מספר אפשרויות לחיבור SPI:

.3 SPI3 - VSPI - בירית המחדל:

- SCLK: GPIO18
 - MOSI: GPIO23
 - MISO: GPIO19
 - CS: GPIO5
- :HSPI - SPI2 .4
- SCLK: GPIO14
 - MOSI: GPIO13
 - MISO: GPIO12
 - CS: GPIO15

מייפוי רגליים:



מערכת החיבור בין RFID-RC522 לברker :ESP32

ESP32	RFID-RC522
Vcc (3.3v)	VCC +3.3V
RST	22
GND	GND
MISO	19
MOSI	23
SCK	18
SS/SDA	5

שימושים נפוצים:

- תקשורת עם צגים
- קריאה/כתיבה לכרטיסי SD
- תקשורת עם חיישנים
- תקשורת עם מודול RF
- העברת נתונים להתקנים היקפיים

להלן דוגמת קוד לשימוש בתקשרות SPI:

```
from machine import Pin, SPI
import time

class SPIExample:
    def __init__(self):
        # הגדלת SPI פינוי
        self.sck = Pin(18, Pin.OUT)      # שעון #
        self.mosi = Pin(23, Pin.OUT)     # מוצא נתונים #
        self.miso = Pin(19, Pin.IN)      # כניסה נתונים #
```

```

        self.cs = Pin(5, Pin.OUT)          בחרית רכיב #

# אתחול אובייקט SPI
self.spi = SPI(2, baudrate=1000000,
               polarity=0, phase=0,
               sck=self.sck,
               mosi=self.mosi,
               miso=self.miso)

def write_data(self, data):
    """ SPI פונקציה לשילוח נתונים דרך
    """
    self.cs.value(0) # הפעלה CS (active low)
    self.spi.write(bytes(data))
    self.cs.value(1) # ביטול CS

def read_data(self, bytes_count):
    """ SPI פונקציה לקריאת נתונים דרך
    """
    self.cs.value(0)
    data = self.spi.read(bytes_count)
    self.cs.value(1)
    return data

def transfer_data(self, data):
    """ פונקציה לשילוח וקבלת בו-זמן נתונים
    """
    self.cs.value(0)
    received = self.spi.write_readinto(bytes(data), bytes(len(data)))
    self.cs.value(1)
    return received

# דוגמת שימוש
def main():
    spi_example = SPIExample()

    # דוגמה לשילוח נתונים #
    data_to_send = [0x55, 0xAA, 0x0F]
    print("שלח נתונים:", data_to_send)
    spi_example.write_data(data_to_send)

    # דוגמה לקריאת נתונים #
    received_data = spi_example.read_data(3)
    print("התבלו נתונים:", list(received_data))

    # דוגמה להעברה דו-כיוונית #
    transfer_data = [0x12, 0x34, 0x56]
    received = spi_example.transfer_data(transfer_data)
    print("נתונים שהעבירו:", transfer_data)
    print("נתונים שהתקבלו:", list(received))

```

```

if __name__ == "__main__":
    main()

```

הסברים נוספים על הקוד:

4. **אתחול SPI:**

- בחירת עroz
- הגדרת קצב השעון (baudrate)
- קביעת הקוטביות והפaza
- הגדרת הפינים

5. **פעולות בסיסיות:**

- write - שליחת נתונים
- read - קריית נתונים
- העברת DO-כיוונית - write_readinto

6. **טיפים לשימוש:**

- תמיד לשחרר את-CS בסיום התקשרות
- להתאים את מהירות השעון לתקן המחבר
- לוודא חיבור נכון של הפינים

להלן מימוש המחלקה המתפל בקורא הcredtisim, יש ליצור קובץ בשם : mfrc522.py ולשמר אותו הקובץ בזיכרון בקר ה-ESP32.

```

from machine import Pin, SPI
import utime
#https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf
class MFRC522:

    OK = const(0)
    NOTAGERR = const(1)
    ERR = const(2)
    REQIDL = const(0x26)
    REQALL = const(0x52)
    AUTHENT1A = const(0x60)
    AUTHENT1B = const(0x61)

    def __init__(self, spi, cs=None, rst=None):
        self.spi = spi
        self.cs = cs
        self.rst = rst
        if self.rst is None:
            self.rst = DummyPin()
        if self.cs is None:
            self.cs = DummyPin()
        self.cs.init(self.cs.OUT, value=1)
        self.rst.init(self.rst.OUT, value=1)
        self.initSensor()

    def initSensor(self):
        self.reset()
        self._wreg(0x2A, 0x8D)
        self._wreg(0x2B, 0x3E)
        self._wreg(0x2D, 30)
        self._wreg(0x2C, 0)

```

```

        self._wreg(0x15, 0x40)
        self._wreg(0x11, 0x3D)
        self.antenna_on()

    def reset(self):
        self._wreg(0x01, 0x0F)

    def antenna_on(self, on=True):
        """
        Turns on the antenna of an MFRC522 RFID module by setting the TxControlReg
        register.
        If the antenna is already on, this method does nothing.
        """
        if on and ~(self._rreg(0x14) & 0x03):
            self._sflags(0x14, 0x03)
        else:
            self._cflags(0x14, 0x03)

    def request(self, mode):
        self._wreg(0x0D, 0x07)
        stat, recv, bits = self._tocard(0x0C, [mode])
        if (stat != self.OK) | (bits != 0x10):
            stat = self.ERR
        return stat, bits

    def anticoll(self):
        ser_chk = 0
        ser = [0x93, 0x20]
        self._wreg(0x0D, 0x00)
        stat, recv, bits = self._tocard(0x0C, ser)
        if stat == self.OK:
            if len(recv) == 5:
                for i in range(4):
                    ser_chk = ser_chk ^ recv[i]
                if ser_chk != recv[4]:
                    stat = self.ERR
            else:
                stat = self.ERR
        return stat, recv

    def select_tag(self, uid):
        """
        Selects a tag or card for communication.
        Args:
            uid (list): The unique identifier of the tag or card.
        Returns:
            int: The status of the command execution (1 or 0).
        """
        buf = [0x93, 0x70] + uid[:5]
        buf += self._crc(buf)
        stat, recv, bits = self._tocard(0x0C, buf)
        return self.OK if (stat == self.OK) and (bits == 0x18) else self.ERR

    def authentication(self, mode, addr, sect, uid):
        return self._tocard(0x0E, [mode, addr] + sect + uid[:4])[0]

    def stop_crypto1(self):
        self._cflags(0x08, 0x08)

```

```

def read(self, addr):
    data = [0x30, addr]
    data += self._crc(data)
    stat, recv, _ = self._tocard(0x0C, data)
    return recv if stat == self.OK else None

def write(self, addr, data):
    buf = [0xA0, addr]
    buf += self._crc(buf)
    stat, recv, bits = self._tocard(0x0C, buf)
    if not (stat == self.OK) or not (bits == 4) or not ((recv[0] & 0x0F) ==
0x0A):
        stat = self.ERR
    else:
        buf = data + self._crc(data)
        stat, recv, bits = self._tocard(0x0C, buf)
        if not (stat == self.OK) or not (bits == 4) or not ((recv[0] & 0x0F) ==
0x0A):
            stat = self.ERR
    return stat

def _wreg(self, reg, val):
    self.cs.value(0)
    self.spi.write(b'%c' % int(0xff & ((reg << 1) & 0x7e)))
    self.spi.write(b'%c' % int(0xff & val))
    self.cs.value(1)

def _rreg(self, reg):
    self.cs.value(0)
    self.spi.write(b'%c' % int(0xff & (((reg << 1) & 0x7e) | 0x80)))
    val = self.spi.read(1)
    self.cs.value(1)
    return val[0]

def _sflags(self, reg, mask):
    self._wreg(reg, self._rreg(reg) | mask)

def _cflags(self, reg, mask):
    self._wreg(reg, self._rreg(reg) & (~mask))

def _tocard(self, cmd, send):
    recv = []
    bits = irq_en = wait_irq = n = 0
    stat = self.ERR
    if cmd == 0x0E:
        irq_en = 0x12
        wait_irq = 0x10
    elif cmd == 0x0C:
        irq_en = 0x77
        wait_irq = 0x30
    self._wreg(0x02, irq_en | 0x80)
    self._cflags(0x04, 0x80)
    self._sflags(0x0A, 0x80)
    self._wreg(0x01, 0x00)
    for c in send:
        self._wreg(0x09, c)
    self._wreg(0x01, cmd)

```

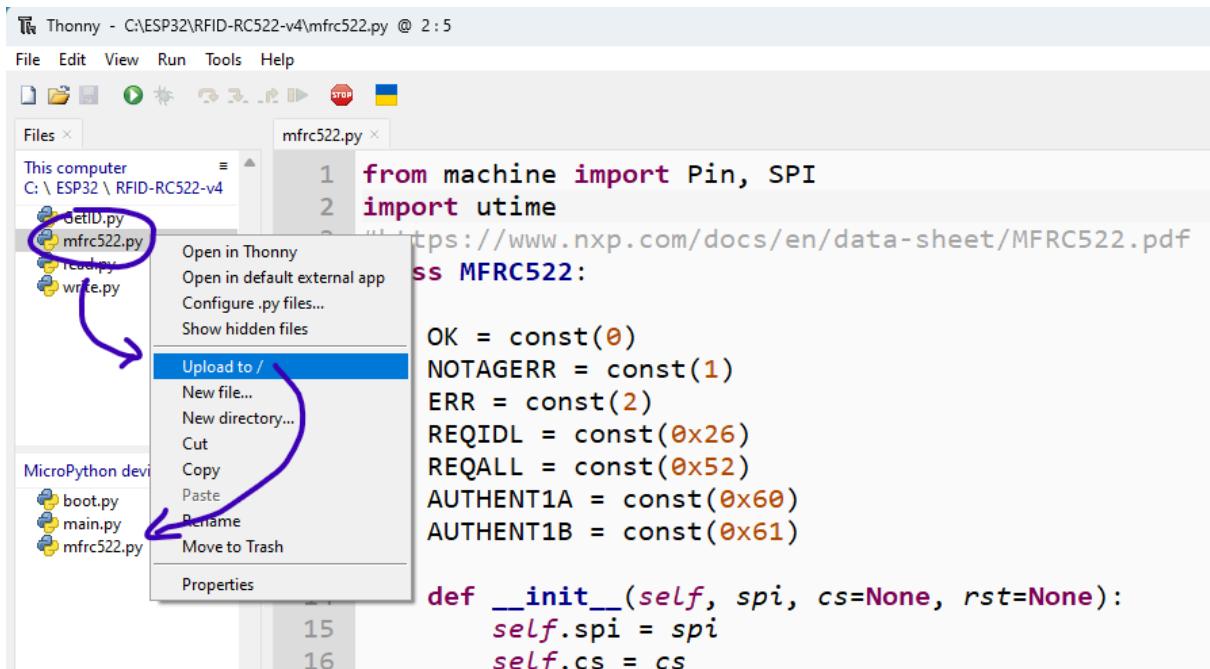
```

if cmd == 0x0C:
    self._sflags(0x0D, 0x80)
i = 2000
while True:
    n = self._rreg(0x04)
    i -= 1
    if ~((i != 0) and ~(n & 0x01) and ~(n & wait_irq)):
        break
self._cflags(0x0D, 0x80)
if i:
    if (self._rreg(0x06) & 0x1B) == 0x00:
        stat = self.OK
        if n & irq_en & 0x01:
            stat = self.NOTAGERR
    elif cmd == 0x0C:
        n = self._rreg(0x0A)
        lbits = self._rreg(0x0C) & 0x07
        if lbits != 0:
            bits = (n - 1) * 8 + lbits
        else:
            bits = n * 8
        if n == 0:
            n = 1
        elif n > 16:
            n = 16
        for _ in range(n):
            recv.append(self._rreg(0x09))
    else:
        stat = self.ERR
return stat, recv, bits

def _crc(self, data):
    self._cflags(0x05, 0x04)
    self._sflags(0x0A, 0x80)
    for c in data:
        self._wreg(0x09, c)
    self._wreg(0x01, 0x03)
    i = 0xFF
    while True:
        n = self._rreg(0x05)
        i -= 1
        if not ((i != 0) and not (n & 0x04)):
            break
    return [self._rreg(0x22), self._rreg(0x21)]

```

ניתן לראות כיצד מעבירים את הקובץ לבקר:



דוגמה 1: קריית קוד הבקר ובניית מנגנון אבטחה בסיסי

להלן הקוד:

```

from mfrc522 import MFRC522
from machine import Pin, SPI, reset

spi = SPI(2,baudrate=100000, polarity=0, phase=0, sck=Pin(18, Pin.OUT),
mosi=Pin(23, Pin.OUT), miso=Pin(19, Pin.IN))
rdr = MFRC522(spi, cs=Pin(5, Pin.OUT), rst=Pin(22, Pin.OUT))
print("Place the card close to the sensor")
print('Presse Ctrl-C to exit')

ApprovedCardList = [
    [58, 58, 199, 36],
    [249, 65, 207, 153]
]

try:
    while True:
        stat, tag_type = rdr.request(rdr.REQIDL)
        if stat == rdr.OK:
            stat, raw_uid = rdr.anticoll()
            if stat == rdr.OK:
                cardId=[raw_uid[0], raw_uid[1], raw_uid[2], raw_uid[3]]
                print("Card detected. id:",cardId)

                for lst in ApprovedCardList:
                    if lst == cardId:
                        print("OK")

except KeyboardInterrupt:
    print('Ctrl-C pressed...exiting')
    reset()

```

דוגמה 2: כתיבת בлок נתונים לזכרון של כרטיס ה-RFID

להלן הקוד:

```
from mfrc522 import MFRC522
from machine import Pin, SPI, reset

spi = SPI(2,baudrate=100000, polarity=0, phase=0, sck=Pin(18, Pin.OUT),
mosi=Pin(23, Pin.OUT), miso=Pin(19, Pin.IN))
rdr = MFRC522(spi, cs=Pin(5, Pin.OUT), rst=Pin(22, Pin.OUT))
print("Place the card close to the sensor")
print('Presse Ctrl-C to exit')

try:
    while True:
        stat, tag_type = rdr.request(rdr.REQIDL)
        if stat == rdr.OK:
            (stat, idCard) = rdr.anticoll()
            if stat == rdr.OK:
                print("Card detected")
                cardId=[idCard[0], idCard[1], idCard[2], idCard[3]]
                print(" - id :", cardId)
                if rdr.select_tag(idCard) == rdr.OK:
                    key = [0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF]
                    DataToSend =
[ord('G'),ord('a'),ord('d'),ord('i'),0,1,2,3,4,5,6,7,8,9,10,255]
                    if rdr.authentication(rdr.AUTHENT1A, 10, key, idCard)
== rdr.OK:
                        stat = rdr.write(10, DataToSend)
                        rdr.stop_crypto1()
                        if stat == rdr.OK:
                            print("Data written to card")
                        else:
                            print("Failed to write data to card")
                    else:
                        print("Authentication error")
                else:
                    print("Failed to select tag")
except KeyboardInterrupt:
    print('Ctrl-C pressed...exiting')
    reset()
```

דוגמה 3: קריית בлок נתונים לזכרון של כרטיס ה-RFID

להלן הקוד:

```
from mfrc522 import MFRC522
from machine import Pin, SPI, reset

spi = SPI(2,baudrate=100000, polarity=0, phase=0, sck=Pin(18, Pin.OUT),
mosi=Pin(23, Pin.OUT), miso=Pin(19, Pin.IN))
rdr = MFRC522(spi, cs=Pin(5, Pin.OUT), rst=Pin(22, Pin.OUT))
print("Place the card close to the sensor")
```

```

print('Presse Ctrl-C to exit')

try:
    while True:
        stat, tag_type = rdr.request(rdr.REQIDL)
        if stat == rdr.OK:
            stat, idCard = rdr.anticoll()
            if stat == rdr.OK:
                print("Card detected")
                cardId=[idCard[0], idCard[1], idCard[2], idCard[3]]
                print(" - id :", cardId)
            if rdr.select_tag(idCard) == rdr.OK:
                key = [255, 255, 255, 255, 255, 255]
                if rdr.authentication(rdr.AUTHENT1A, 10, key, idCard) ==
rdr.OK:
                    print("Read data from card: %s" % rdr.read(10))
                    rdr.stop_crypto1()
                else:
                    print("Authentication error")
            else:
                print("Failed to select tag")
except KeyboardInterrupt:
    print('Ctrl-C pressed...exiting')
    reset()

```

משימה 21 - עדכון RTC פנימי בברר תוך שימוש ב- API מבוסס JSON

קישורים:

<https://www.hackster.io/alankrantas/very-simple-micropython-esp8266-esp-12-web-clock-3c5c6f>

<https://docs.micropython.org/en/latest/esp32/quickref.html#real-time-clock-rtc>

<https://techtutorialsx.com/2017/06/11/esp32-esp8266-micropython-http-get-requests/>

<https://docs.micropython.org/en/latest/library/ujson.html>

<http://docs.micropython.org/en/v1.9.3/pyboard/library/pyb.RTC.html#pyb.RTC.datetime>

רכיב ESP32 כולל רכיב RTC פנימי, להלן דוגמת קוד המאחזרת את התאריך והשעה מהרכיב:

```
from machine import RTC

rtc = RTC()
print("Current time:" + str(rtc.datetime()))

(year, month, day, wday, hrs, mins, secs, subsecs) = rtc.datetime()
print("Time: ",hrs,":", mins,":",secs, sep="")
print("Date: ",day,"/", month,"/",year, sep="")
```

נקבל את הפלט הבא:

```
MPY: soft reboot
Already connected
('10.0.0.20', '255.255.255.0', '10.0.0.138', '10.0.0.138')
Current time:(2024, 10, 21, 0, 18, 21, 29, 686410)
Time: 18:21:29
Date: 21/10/2024
```

>>>

עדכון שעה

במטרה לעדכן את השעה באופן אוטומטי תורש שימוש ביכולת הגישה של הברר לאינטרנט נעבור על השלבים הבאים:

תחילה נדגים כיצד חיבור לאינטרנט יעזר לנו לקבל עדכון זמן.

כנסו לכתובות הבא:

<http://worldtimeapi.org/api/timezone/Asia/Jerusalem>

קישור זה מחזיר לנו מחרוזת טקסט בפורמט JSON הכוללת השעה המדויקת באופן הבא:

The screenshot shows a browser window with the URL worldtimeapi.org/api/timezone/Asia/Jerusalem. The page displays a JSON object with the following structure:

```
{
  "utc_offset": "+03:00",
  "timezone": "Asia/Jerusalem",
  "day_of_week": 1,
  "day_of_year": 295,
  "datetime": "2024-10-21T14:49:25.694932+03:00",
  "utc_datetime": "2024-10-21T11:49:25.694932+00:00",
  "unixtime": 1729511365,
  "raw_offset": 7200,
  "week_number": 43,
  "dst": true,
  "abbreviation": "IDT",
  "dst_offset": 3600,
  "dst_from": "2024-03-29T00:00:00+00:00",
  "dst_until": "2024-10-26T23:00:00+00:00",
  "client_ip": "77.137.23.74"
}
```

נבחן את מבנה הנתונים שהתקבל:

```
{
  "utc_offset": "+03:00",
  "timezone": "Asia/Jerusalem",
  "day_of_week": 1,
  "day_of_year": 295,
  "datetime": "2024-10-21T14:49:25.694932+03:00",
  "utc_datetime": "2024-10-21T11:49:25.694932+00:00",
  "unixtime": 1729511365,
  "raw_offset": 7200,
  "week_number": 43,
  "dst": true,
  "abbreviation": "IDT",
  "dst_offset": 3600,
  "dst_from": "2024-03-29T00:00:00+00:00",
  "dst_until": "2024-10-26T23:00:00+00:00",
  "client_ip": "77.137.23.74"
}
```

JSON - JavaScript Object Notation הוא פורמט קל לשאול לאחסון והחלפת נתונים. הוא קל לקרוא ולכתחה
עבור בני אדם, וקל לניטוח וליצירה עבור מחשבים. SONJSON מבוסס על שני מבנים:

1. אוסף של זוגות שם/ערך (בדומה לאובייקט, מילון, או מערך אוטוציאיטיבי)
2. רשימה מסודרת של ערכים (בדומה לערך או רשימה)

להלן דוגמה בסיסית של קובץ SONJSON:

```
{
  "שם": "ישראל ישראלוו",
  "גיל": 30,
```

```

        "עיר": "תל אביב",
        "בשווי": false,
        ["תחביבים": ["קריאת", "טיולים", "צילום"],
        {"עובדה":
            "פקיד": " מהנדס תוכנה",
            "חברה": "ABC"
        }
    }
}

```

הסביר התוכן:

- המבנה הכללי הוא אובייקט, מיוצג על ידי סוגרים מסווגלים `{}'.
- בתוך האובייקט יש זוגות של שם:ערך, מופרדים בפסיקים.
- השמות תמיד תמיד במרכאות כפולות.
- הערכים יכולים להיות:
 - מחזוזות (במרכאות כפולות)
 - מספרים
 - `true` או `false` (ערכים בוליאניים)
 - `null` -
 - מערך (מיוצג בסוגרים מרובעים `[]`)
 - אובייקט אחר (מיוצג בסוגרים מסווגלים `{}'`)

להלן דוגמת קוד ב-*MicroPython* המדגימה כיצד לקרוא ולפענח קובץ JSON:

```

import ujson

# JSON קריית קובץ
def read_json_file(filename):
    try:
        with open(filename, 'r') as file:
            data = file.read()
        return ujson.loads(data)
    except OSError as e:
        print("שגיאה בקריאת הקובץ", e)
        return None

# JSON-פונCTION והדפסת תוכן ה
def parse_and_print_json(json_data):
    if json_data is not None:
        print("שם:", json_data.get("שם"))
        print("גיל:", json_data.get("גיל"))
        print("עיר:", json_data.get("עיר"))
        print("לא" " נשוי:", "כן" "if json_data.get(" נשוי") else)

        print("תחביבים:")
        for hobby in json_data.get("תחביבים", []):

```

```

        print("- " + hobby)

    work = json_data.get("עובדה", {})
    print("עובדה:")
    print("תפקיד:", work.get("תפקיד"))
    print("חברה:", work.get("חברה"))

# שימוש בפונקציות
filename = "data.json"
json_data = read_json_file(filename)
parse_and_print_json(json_data)

```

קוד זה מדגים כיצד לקרוא קובץ JSON, לפענה אותו, ולגשת נתונים בצורה בטוחה ויעילה ב-MicroPython.

נקבל את הפלט הבא:

The screenshot shows the Thonny IDE interface. On the left, the file tree shows a folder structure: 'This computer' contains 'C:\ESP32\AllBookFiles\21_RTC' which includes 'data.json', 'ReadJSON.py', and 'time.json'. Below that is 'MicroPython device' with files 'boot.py', 'data.json' (circled in blue), 'lil9341.py', 'MicroPython128x128.raw', and 'xpt2046.py'. The main window displays the Python code for reading JSON. The 'Shell' tab at the bottom shows the program's output: 'MPY: soft reboot', 'Already connected', and a series of printed JSON objects. Handwritten annotations in blue curly braces group the printed objects into categories: 'שמות', 'גיל', 'עיר', 'נשי', 'כן', and 'תפקיד'.

```

import ujson
# קובץ קרייתם
def read_json_file(filename):
    try:
        with open(filename, 'r') as file:
            data = file.read()
        return ujson.loads(data)
    except OSError as e:
        print(e)
        print("הקובץ בקריאת שגיאה")
        return None

# תומך קובץ וודפסת פונק
def parse_and_print_json(json_data):
    if json_data is not None:
        print("שם:", json_data.get("שם"))
        print("גיל:", json_data.get("גיל"))
        print("עיר:", json_data.get("עיר"))
        print("נשי:", "כן" if json_data.get("נשי") else "לא")

```

```

MPY: soft reboot
Already connected
('10.0.0.20', '255.255.255.0', '10.0.0.138', '10.0.0.138')
שם: ישראל ישראלי
גיל: 30
עיר: תל אביב
נשי: לא
תפקיד:
- קריירה -
- טכנולוגיות -
- ציילום -
עבדה:
- מהנדס תוכנה -
- חברה: טכנולוגיות ABC

```

הסביר הקוד:

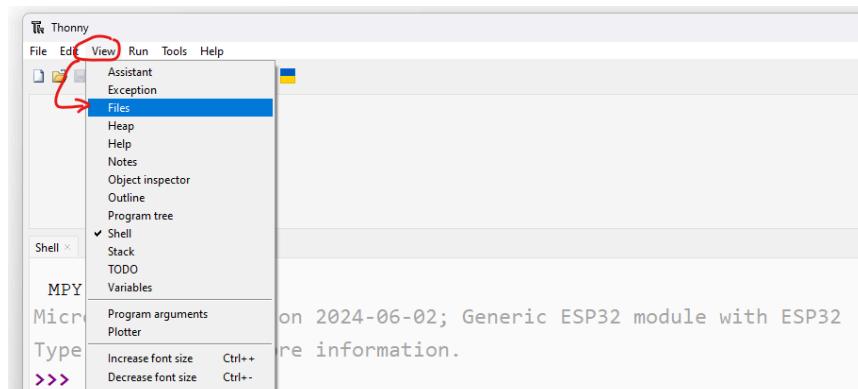
1. אנו משתמשים במודול `json` של MicroPython לטיפול ב-JSON.
2. הפונקציה `read_json_file` קוראת את תוכן הקובץ ומשתמשת ב-`ujson.loads` כדי לפענה את ה-JSON מבנה נתונים של Python.
3. הפונקציה `parse_and_print_json` מקבלת את הנתונים המפוענחים ומדפיסה אותם בצורה מסודרת.

4. אנו משתמשים בשיטה `get` כדי לגשת לערכים ב-`NODE`, מה שמאפשר לנו לטעל במקרים שבהם מפתח מסויים אינו קיים.

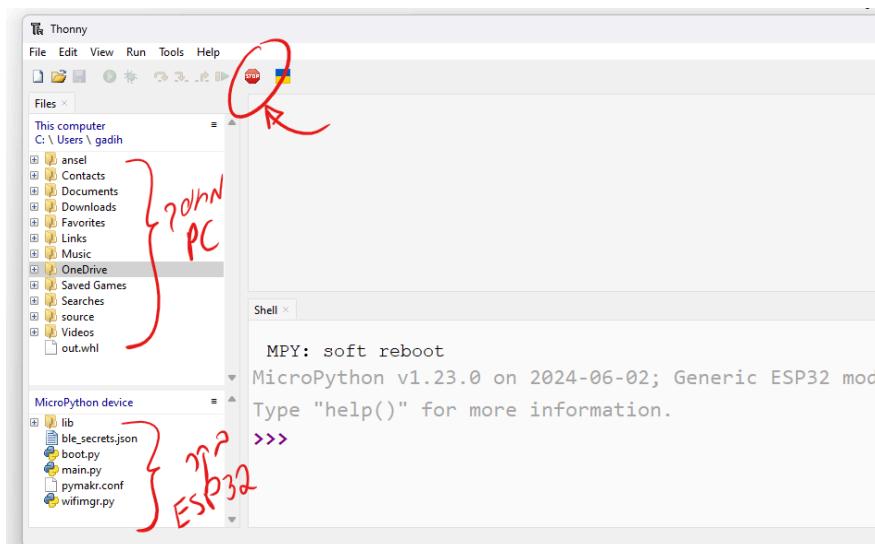
בתרגיל זה נחלץ את השעה מຕוך `datetime` ונדען בעזרתו את השעון הפנימי של הבקר.

שלב 1: חיבור הבקר למחשב

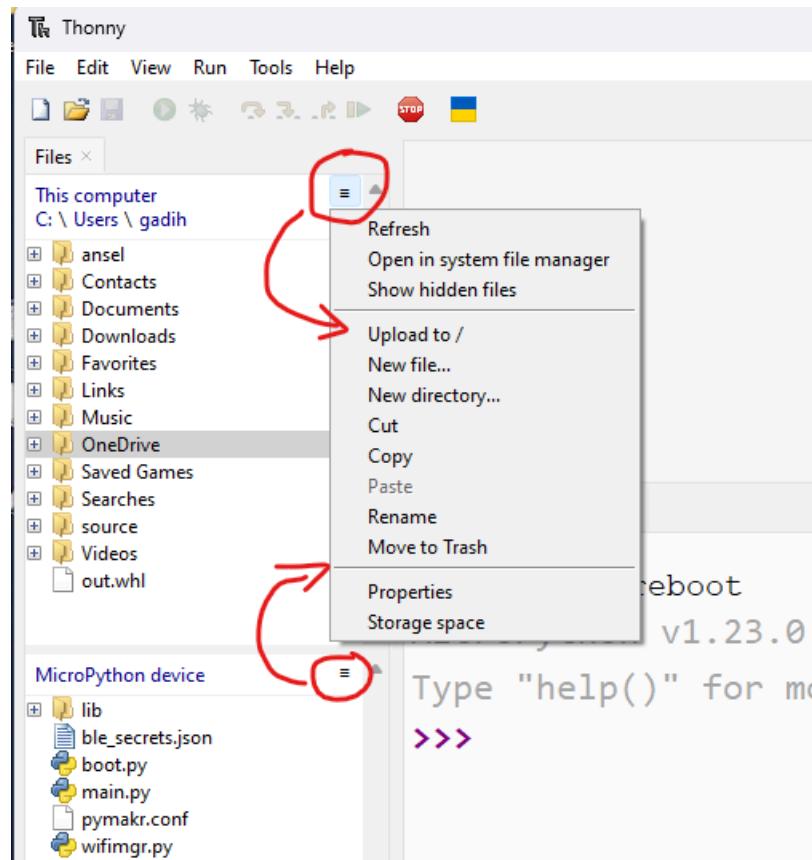
בקר ESP32 מצוייד בזכרון הבזק בנפח של 4 מגה-בייט שבו ניתן לשמר קבועים ונתונים. נעזר בסביבת הפיתוח `Thonny` כדי לבדוק את הקבצים שבזיכרון הבקר מיד לאחר אתחול בקר בקורסחה חדשה. נעשה זאת על ידי לחיצה על `View` ו- `File` כדי לראות את הקבצים שבבקר:



נפתח לנו המסר הבא:



בעזרת החלון שנפתח לנו יכולים לראות אילו קבצים שמורים בזכרון של הבקר. כמו כן ניתן לבצע את כל הפעולות הבסיסיות על הקבצים כמו הוספה קבצים, מחיקת קבצים, ייצור ומחיקה של תיקיות ושינוי של של קובץ.



נעזר במכשיר שפתחנו כדי להעתיק את הקובץ `boot.py` מהזיכרון של הבקר למחשב על ידי ליצחה על `Upload to /`

להלן קוד התוכנית [עבור הקובץ boot.py](#):

```
# This file is executed on every boot (including wake-boot from deepsleep)
import network

def connect():
    ssid = "yourNetworkName"
    password = "yourNetworkPassword"

    station = network.WLAN(network.STA_IF)

    if station.isconnected() == True:
        print("Already connected")
        print(station.ifconfig())
        return

    station.active(True)
    station.connect(ssid, password)

    while station.isconnected() == False:
        pass

    print("Connection successful")
    print(station.ifconfig())
```

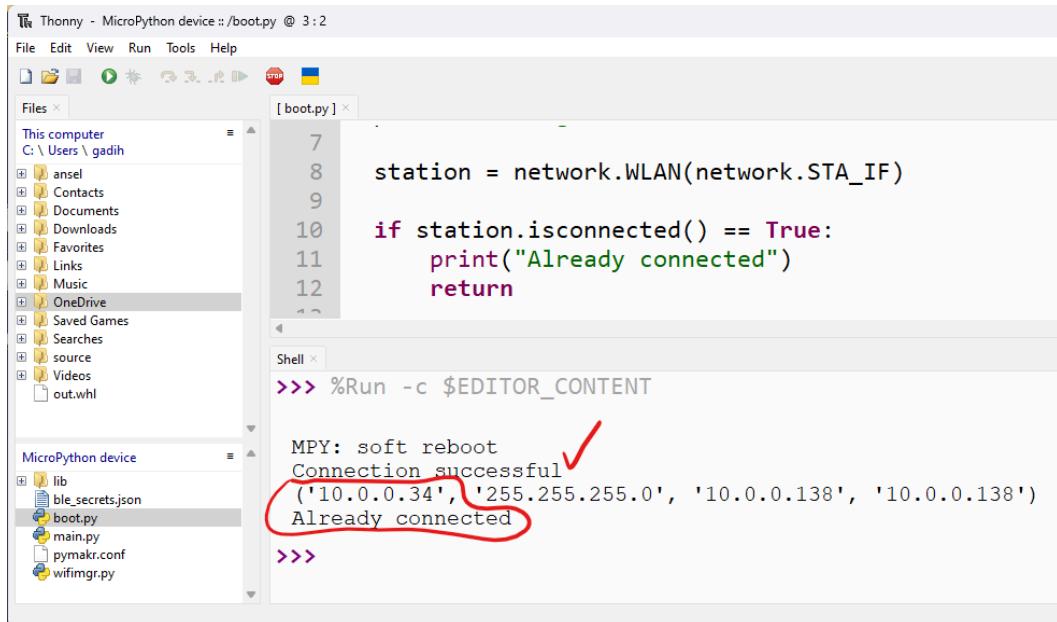
```
connect()
```

חשוב: עדכנו בקובץ את שם רשת ה- WiFi הזמין והסימה במקום השירות הבא:

```
ssid = "yourNetworkName"  
password = "yourNetworkPassword"
```

לאחר כתיבת הקובץ boot.py נעלם אותו לבקר.

כדי לבדוק שאכן הבקר התחבר לאינטרנט נבצע אתחול ונקבל את הפלט הבא:



בעקבות חיבורית מוצלחת נקלט בחילון ה- Terminal את מאפייני החיבור שלהם:

- כתובת IP של הבקר
- ערך ה- subnet mask
- כתובת ה- gateway
- כתובת ה- DNS

שלב 2: כתיבת קבצי הקוד לקריאת השעה

להלן קוד התוכנית:

```
from machine import RTC, reset  
import ntptime  
from time import localtime, sleep  
  
UTC_OFFSET = 3 # UTC+3:00  
  
# Create an independent clock object  
rtc = RTC()  
  
print(rtc.datetime())
```

```

# Get UTC time from NTP server (pool.ntp.org) and store it to internal RTC
ntptime.settime()

# Display UTC (Coordinated Universal Time / Temps Universel Coordonné)
(year, month, day, wday, hrs, mins, secs, subsecs) = rtc.datetime()
print(f"UTC Time: {year}-{month}-{day} {hrs}:{mins}:{secs}")

# Get epoch time in seconds (for timezone update)
sec = ntptime.time()

# Update your epoch time in seconds and store in to internal RTC
sec = int(sec + UTC_OFFSET * 60 * 60)
(year, month, day, hrs, mins, secs, wday, yday) = localtime(sec)
rtc.datetime((year, month, day, wday, hrs, mins, secs, 0))

print(f"Local RTC time: UTC+{UTC_OFFSET}:00")

try:
    while True:
        # Read values from internal RTC
        (year, month, day, wday, hrs, mins, secs, subsecs) = rtc.datetime()
        print(f"{year}-{month}-{day} {hrs}:{mins}:{secs}")

        # Delay 30 seconds
        sleep(30)

except KeyboardInterrupt:
    print('Ctrl-C pressed... exiting')
    reset()

```

נקבל פלט הדומה לזה המוצג כאן:

```

Shell >>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Already connected
('10.0.0.20', '255.255.255.0', '10.0.0.138', '10.0.0.138')
(2024, 10, 21, 0, 18, 29, 19, 579591)
UTC Time: 2024-10-21 15:29:20
Local RTC time: UTC+3:00
2024-10-21 18:29:20
2024-10-21 18:29:50

```

להלן דוגמת קוד, שפוטה הכוללת התחברות לאינטרנט דרך WiFi, קבלת שעה משרות באינטרנט והציגת השעה המעודכנת, הכל באותו קוד.

```
import ntptime
from machine import RTC
import network
from time import sleep

rtc=RTC()
ssid='_____XXX_____'
password='_____XXX_____'
station=network.WLAN(network.STA_IF)
station.active (True)
station.connect(ssid,password)
print("Connecting to wifi: ",end="")
while station.isconnected()==False:
    print('. ',end="")
    sleep (0.5)
print ("Connected")
ntptime.host="pool.ntp.org"
ntptime.settime()
while True:
    datetime=rtc.datetime()
    year=str(datetime[0])
    month=str(datetime[1])
    daymonth=str(datetime[2])
    dayweek=str(datetime[3])
    hour=str(datetime[4]+2)
    minute=str(datetime[5])
    second=str(datetime[6])
    print ("date: "+daymonth+"/"+month+"/"+year)
    print ("time: "+hour+":"+minute+":"+second)
    sleep (0.5)
```

*** תודה ליאוב גולן על הקוד.

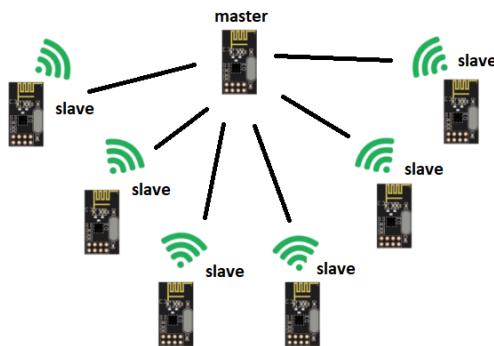
משימה 22 - תקשורת אלחוטית מבוססת מקמ"ש NRF24L01

קישורים:

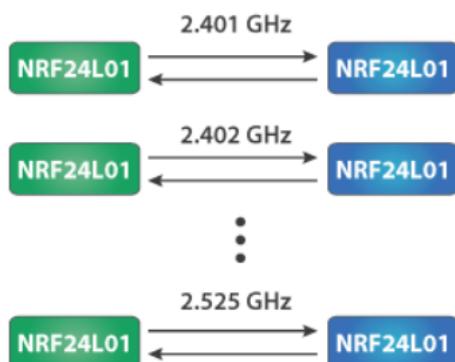
<https://github.com/micropython/micropython/tree/master/drivers/nrf24l01>

<http://docs.micropython.org/en/v1.9.3/pyboard/library/pyb.SPI.html>

مصدر מקלט NRF24L01 עובד בתדר של 2.4 גיגה הרץ. הרכיב מסוגל להעביר תקשורת ספרטיטית בקצב שבין עד 250Kbps 2Mbps. בטוחה שנע בין מספר מטרים בחלל סגור ועד 100 מטר בחלל פתוח. מקמ"ש (مصدر/מקלט) NRF24L01 יכול להשתמש בו-125 ערוצי תקשורת שונים. כאשר כל ערוץ יכול לתקשר בו-זמנית בין יחידה מרכזית אחת יחד עם עד 6 יחידות משנה נוספת.



כל ערוץ תקשורת מקבל תדר פנוי בתחום שבין 2.401GHz עד 2.525GHz



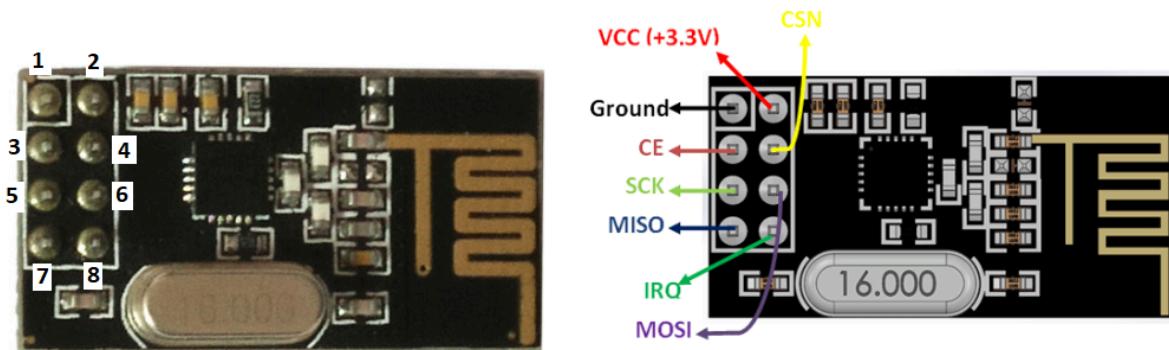
להלן רשימת המאפיינים הטכניים של הרכיב

- 2.4GHz RF transceiver Module
- Operating Voltage: 3.3V
- Nominal current: 50mA
- Range : 50 – 200 feet
- Operating current: 250mA (maximum)
- Communication Protocol: SPI
- Baud Rate: 250 kbps - 2 Mbps.
- Channel Range: 125
- Maximum Pipelines/node : 6
- Low cost wireless solution

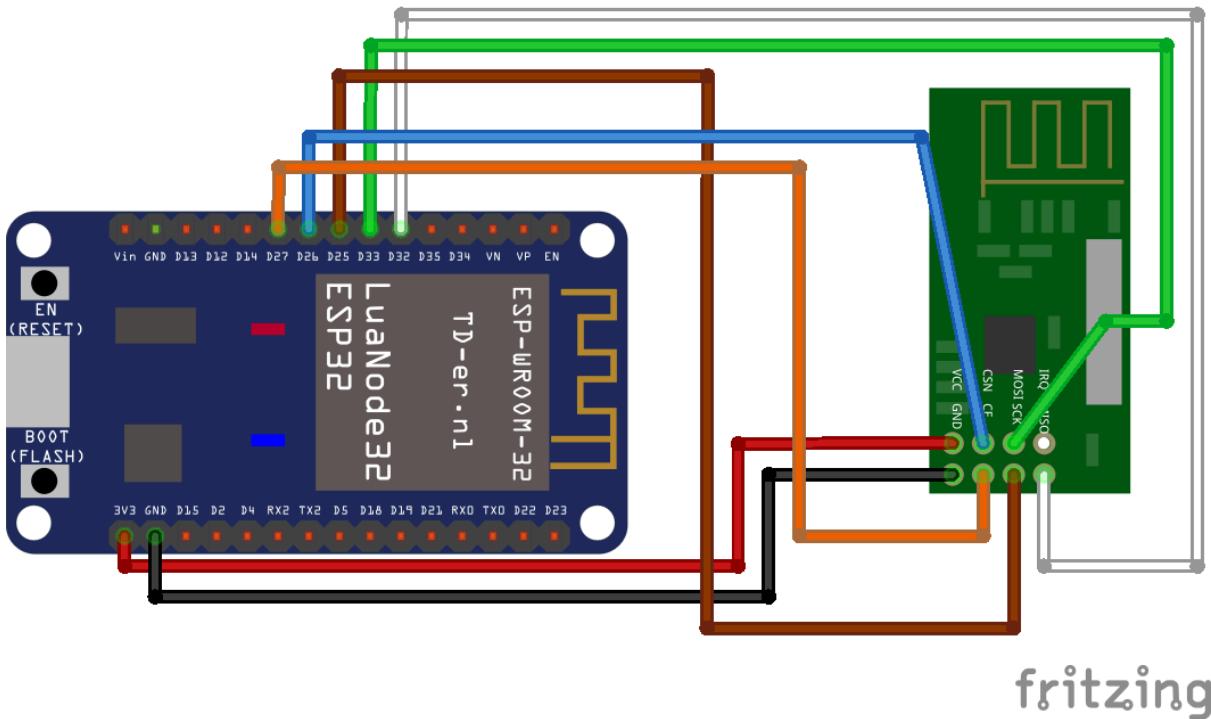
שימוש לבן מקום"ש עובד במתוח עבודה של 3.3V ולא 5V. למרות זאת הדק' הבדיקה של הרכיב יכולם להתחבר לבקרים העובדים בرمות של 5V כמו Arduino UNO.

מיופי רגליים:

- 1 - Ground - Connected to the Ground of the system
- 2 - Vcc - Powers the module using 3.3V
- 3 - CE - Used to enable SPI communication
- 4 - CSN - This pin has to be kept high always, else it will disable the SPI
- 5 - SCK - Provides the clock pulse using which the SPI communication works
- 6 - MOSI - Connected to MOSI pin of MCU, for the module to receive data from the MCU
- 7 - MISO - Connected to MISO pin of MCU, for the module to send data from the MCU
- 8 - IRQ - It is an active low pin and is used only if interrupt is required



אוף החיבור בין בקר ESP32 לבין מוקם"ש NRF24L01:



fritzing

תקשורת SPI בברker ESP32

There are two SPI drivers. One is implemented in software (bit-banging) and works on all pins, and is accessed via the `machine.SPI` class.

There are two hardware SPI channels that allow faster transmission rates (up to 80Mhz). These may be used on any IO pins that support the required direction and are otherwise unused (see Pins and GPIO) but if they are not configured to their default pins then they need to pass through an extra layer of GPIO multiplexing, which can impact their reliability at high speeds. Hardware SPI channels are limited to 40MHz when used on pins other than the default ones listed below.

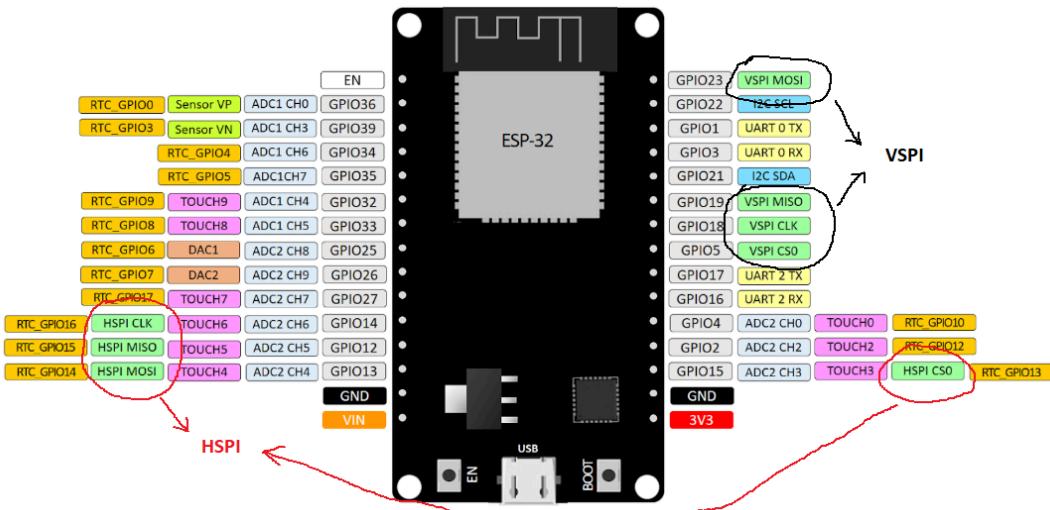
	HSPI (id=1)	VSPI (id=2)
sck	14	18
mosi	13	23
miso	12	19

Hardware SPI has the same methods as Software SPI above:

```
from machine import Pin, SPI

hspi = SPI(1, 10000000, sck=Pin(14), mosi=Pin(13), miso=Pin(12))

vspi = SPI(2, baudrate=80000000, polarity=0, phase=0, bits=8, firstbit=0,
           sck=Pin(18), mosi=Pin(23), miso=Pin(19))
```



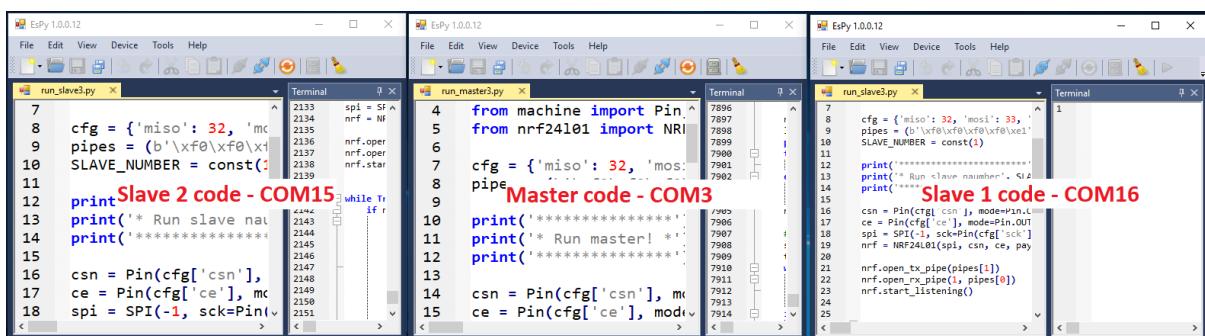
בתרגום זה נעשה שימוש ב-SPI_7 כאשר נשנה את הגדרות הדק' התקשרות שבין הבקר למקם"ש על ידי תוכנה. להלן הוראת האתחול של תקשורת SPI בקוד תוך כדי הגדרת הדק' העובדה:

```
vsopi = SPI(2, baudrate=2000000, polarity=0, phase=0, bits=8, firstbit=0,
sck=Pin(18), mosi=Pin(23), miso=Pin(19))
```

במשימה זו נפתח קוד תוכנה המיציר תקשורת נתונים מבוססת טקסט בין יחידת בוס (Master) לבין 2 יחידות משנה (Slave).

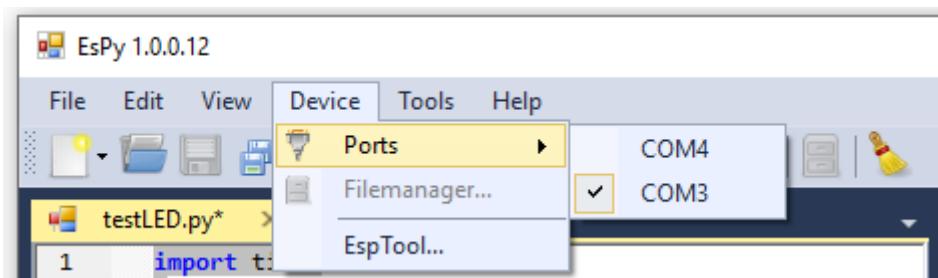
במשימה נעשה שימוש בשלוש בקרים ESP32 המחברות כל אחת למקום"ש NRF24L01 ועובדים בו זמנית. אחד מהם מוגדר בתוכנה כ-Master ו-2 האחרים יוגדרו בתוכנה-C-Slave.

כדי למש את התרגיל נפתח במקביל 3 סביבות פיתוח על גבי המחשב ובכל אחת מהם נתחבר לבקר אחר בהתאם למפתח שאלוי מחוברים כל אחד מהבקרים.

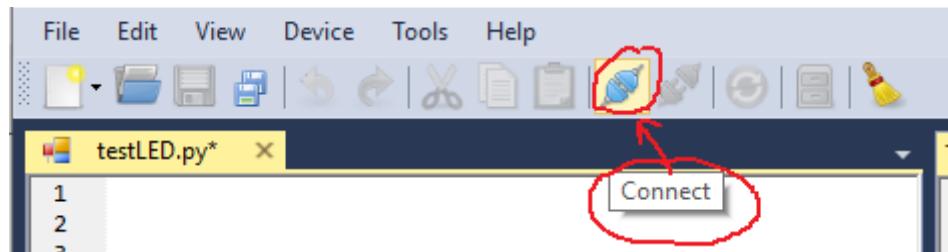


שלב 1: חיבור שלוש בקרים למחשב אחד

נפעיל את קובץ ההרצה EsPy.exe שלוש פעמים על אותו המחשב ונתחבר לכל אחד מהבקרים על ידי לחיצה על Port - Device



ואז נלחץ על Connect

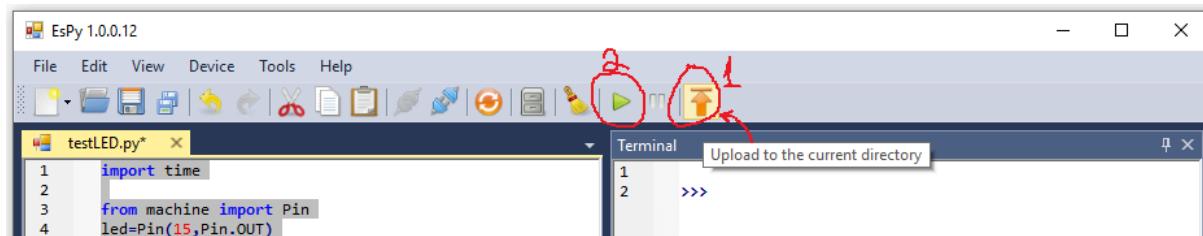


נפתח קובץ Python חדש בכל אחד מהບקרים כדי להעתיק לבקר את קוד המחלקה `radio.NRF2401`.

גרסה עדכנית של המחלקה ניתן להוריד דרך הקישור הבא:

<https://github.com/micropython/micropython-lib/blob/master/micropython/drivers/radio/nrf24l01/nrf24l01.py>

נצרוב את הקובץ שפתחנו על כל אחד משלוש הבקרים על שימוש בלחצן "Upload to the current directory"



שלב 2: כתיבת קבצי הקוד

כתבו 3 קודים שונים האחד לבקר שיממש את השרת (Master) ו-2 קודים ל-2 תוכנות Save שימתינו לפניו מהשרת.

להלן קוד התוכנית המיועדת ל-Master :

```
import sys
import ustruct as struct
import utime
from machine import Pin, SPI
from nrf24l01 import NRF24L01
```

```

cfg = {'miso': 32, 'mosi': 33, 'sck': 25, 'csn': 26, 'ce': 27}
pipes = (b'\xf0\xf0\xf0\xf0\xe1', b'\xf0\xf0\xf0\xf0\xd2')

print('*****')
print('* Run master! *')
print('*****')

csn = Pin(cfg['csn'], mode=Pin.OUT, value=1)
ce = Pin(cfg['ce'], mode=Pin.OUT, value=0)
spi = SPI(-1, sck=Pin(cfg['sck']), mosi=Pin(cfg['mosi']),
miso=Pin(cfg['miso']))

nrf = NRF24L01(spi, csn, ce, payload_size=16)

nrf.open_tx_pipe(pipes[0])
nrf.open_rx_pipe(1, pipes[1])
nrf.start_listening()

num_successes = 0
num_failures = 0
led_state = 0

while num_successes < 10 and num_failures < 10:

    nrf.stop_listening()      #stop listening and send packet
    lst =[utime.ticks_ms(),17,18,19]
    print('Sending:', lst)
    try:
        nrf.send(struct.pack('iiii', lst[0],lst[1],lst[2],lst[3]))
    except OSError:
        pass

    nrf.start_listening()      #start listening again

    # wait for response
    start_time = utime.ticks_ms()
    timeout = False

```

```

while not nrf.any() and not timeout:
    if utime.ticks_diff(utime.ticks_ms(), start_time) > 250:
        timeout = True

if timeout:
    print('Failed, response timed out')
    num_failures += 1

else:
    rlst = struct.unpack('ii', nrf.recv())
    print('Got OK response from slave', rlst[1], 'in',
utime.ticks_diff(utime.ticks_ms(), rlst[0]), 'ms')
    num_successes += 1

utime.sleep_ms(250)

print('Master finished sending; successes=%d, failures=%d' %
(num_successes, num_failures))

```

להלן קוד התוכנית המיעדרת לכל אחד מ-2 ה-Slave:

```

import sys
import ustruct as struct
import utime
from machine import Pin, SPI
from nrf24l01 import NRF24L01
from micropython import const

cfg = {'miso': 32, 'mosi': 33, 'sck': 25, 'csn': 26, 'ce': 27}
pipes = (b'\xf0\xf0\xf0\xf0\xe1', b'\xf0\xf0\xf0\xf0\xd2')
SLAVE_NUMBER = const(1)

print('*****')
print('* Run slave naumber', SLAVE_NUMBER, '*')
print('*****')

csn = Pin(cfg['csn'], mode=Pin.OUT, value=1)

```

```

ce = Pin(cfg['ce'], mode=Pin.OUT, value=0)

spi = SPI(-1, sck=Pin(cfg['sck']), mosi=Pin(cfg['mosi']),
miso=Pin(cfg['miso']))

nrf = NRF24L01(spi, csn, ce, payload_size=16)

nrf.open_tx_pipe(pipes[1])
nrf.open_rx_pipe(1, pipes[0])
nrf.start_listening()

while True:

    if nrf.any():

        while nrf.any():

            buf = nrf.recv()
            lst = struct.unpack('iiii', buf)
            print('received:', lst[0], lst[1], lst[2], lst[3])
            utime.sleep_ms(15)

        # Give master time to get into receive mode.
        utime.sleep_ms(10)
        nrf.stop_listening()
        try:
            nrf.send(struct.pack('ii', lst[0], SLAVE_NUMBER))
        except OSError:
            pass
        print('sent response')
        nrf.start_listening()

```

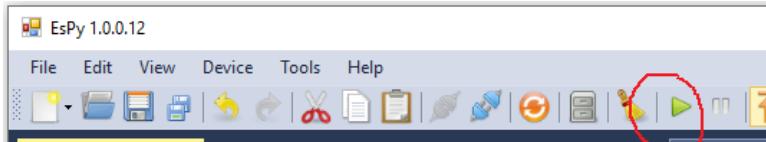
ההבדל היחיד בין 2 תוכנות ה-Slave הוא מספרו כפי שמוגדר בשורה הבאה:

```
SLAVE_NUMBER = const(1)
```

הגדירו לכל אחד מתוכנות ה-Slave מספר שונה.

צרוב את כל אחד מהקבצים שיצרנו על שימוש בלחץ "Upload to the current directory"

להרצת התוכנית בברouter נழמוד על הקובץ שכתבנו ונלחץ על Run



חשיבות: תוכנות ה-Slave חייבות לזרז תחילת לפני הריצת תוכנת ה-Master. זה בגלל שתוכנת ה-Master היא זו שיזמת תקשורת עם ה-Slave. והאחרונה (תוכנת ה-Slave) מוחילה לו - Master הודעת אישור. ללא תוכנת .Failed, response timed out נקבל הודעה Slave

נקבל פלט הדומה לזה המוצג כאן:

```

run_master3.py
1 led_state = 0
2
3 while num_successes < 10 and num_failures < 10:
4     nrf.stop_listening()      #stop listening and send
5     lst =[utime.ticks_ms(),17,18,19]
6     print('Sending:', lst)
7     try:
8         nrf.send(struct.pack('iiii', lst[0],lst[1],1
9     except OSError:
10        pass
11
12    nrf.start_listening()    #start listening again
13
14    # wait for response
15

Port: COM3 File: C:\MicroPython\DE\MySourceCode\NRF24L01\run_master3.py
Terminal
***** * Run master! * *****
8023 Sending: [29691506, 17, 18, 19]
8024 Got OK response from slave 1 in 42 ms
8025 Sending: [29691800, 17, 18, 19]
8026 Got OK response from slave 1 in 40 ms
8027 Sending: [29692092, 17, 18, 19]
8028 Got OK response from slave 1 in 40 ms
8029 Got OK response from slave 1 in 40 ms
8030 Sending: [29692385, 17, 18, 19]
8031 Got OK response from slave 1 in 40 ms
8032 Sending: [29692676, 17, 18, 19]
8033 Got OK response from slave 1 in 39 ms
8034 Sending: [29693551, 17, 18, 19]
8035 Got OK response from slave 1 in 40 ms
8036 Sending: [29692968, 17, 18, 19]
8037 Got OK response from slave 1 in 40 ms
8038 Sending: [29693260, 17, 18, 19]
8039 Got OK response from slave 1 in 39 ms
8040 Sending: [29693843, 17, 18, 19]
8041 Got OK response from slave 1 in 40 ms
8042 Sending: [29693843, 17, 18, 19]
8043 Got OK response from slave 1 in 39 ms
8044 Sending: [29694134, 17, 18, 19]
8045 Got OK response from slave 1 in 40 ms
8046 Master finished sending; successes=10, failures=0
8047

run_slave3.py
1 import sys
2 import ustruct as struct
3 import utime
4 from machine import Pin, SPI
5 from nrf24l01 import NRF24L01
6 from micropython import const
7
8 cfg = {'miso': 32, 'mosi': 33, 'sck': 25, 'csn': 26, 'ce':
9 pipes = (b'\xf0\xf0\xf0\xe1', b'\xf0\xf0\xf0\xd2')
10 SLAVE_NUMBER = const(1)
11
12 print('*****')
13 print('* Run slave naumber', SLAVE_NUMBER, '*')
14 print('*****')

Port: COM3 File: C:\MicroPython\DE\MySourceCode\NRF24L01\run_slave3.py
Terminal
***** * Run slave naumber 1 * *****
2247 received: 29691506 17 18 19
2248 sent response
2249 received: 29691800 17 18 19
2250 sent response
2251 received: 29692092 17 18 19
2252 sent response
2253 received: 29692385 17 18 19
2254 sent response
2255 received: 29692676 17 18 19
2256 sent response
2257 received: 29693551 17 18 19
2258 sent response
2259 received: 29693843 17 18 19
2260 sent response
2261 received: 29694134 17 18 19
2262 sent response
2263 received: 29693260 17 18 19
2264 sent response
2265 received: 29693551 17 18 19
2266 sent response
2267 received: 29693843 17 18 19
2268 sent response
2269 received: 29694134 17 18 19
2270 sent response

```

משימה 23 - חישוני משקל המבוסס על ממיר HX711

קישורים:

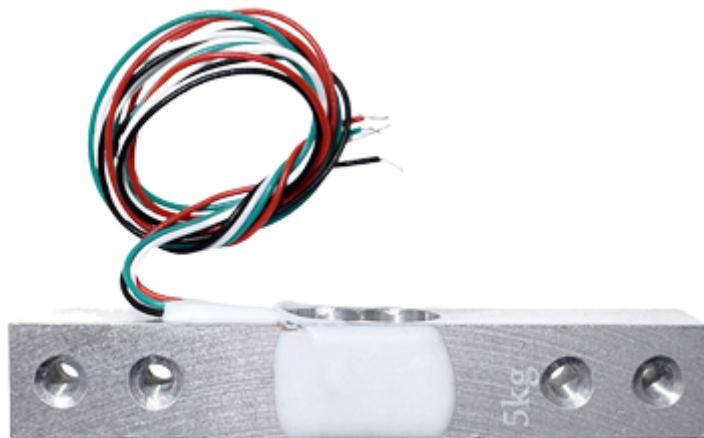
<https://www.arikporat.com/wp-content/uploads/2024/03/%D7%97%D7%99%D7%99%D7%A9%D7%9F-%D7%9E%D7%A9%D7%A7%D7%9C-2.pdf>

<https://github.com/SergeyPiskunov/micropython-hx711>

נסביר תחילה את נושא חישוני המשקל והממיר HX711

עקרון פעולה Load Cell

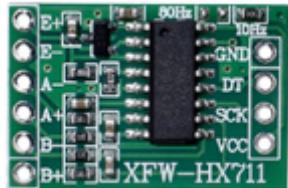
הא חישון המבוסס על גשר ויטסטון (Wheatstone Bridge) המורכב מ-4 נגדים רגיסטרים למאץ (Strain Gauges). כאשר מפעילים כוח על ה-Load Cell, המבנה המכני שלו מתעוזת במקצת וגורם לשינוי בתנגדות של הנגדים. השינוי בהתנגדות יוצר הפרש מתח קטן מאוד (בסדר גודל של מיקרו-וואולטים) בין שתי נקודות בمعالג הגשר.



עקרון פעולה HX711

hx711 הוא ממיר אנלוגי-לדיגיטלי (ADC) בעל דיק גובה של 24 ביט המתוכנן במיוחד עבור חישוני משקל. הוא מכיל:

- מגבר אות דיפרנציאלי בעל הגבר מתוכנת של 128 או 64
- ממיר ADC בעל רזולוציה גבוהה
- מעגלי עיבוד אות פנימיים
- ממשק תקשורת טורי פשוט



חיבור Load Cell ל-HX711

החיבור הבסיסי כולל:

1. E+/VCC מתח הזנה חיובי
2. E-/GND GND
3. A+/OUT+ אות חיובי דיפרנציאלי
4. A-/OUT- אות שלילי דיפרנציאלי

פרוטוקול התקשרות

התקשרות בין ESP32 ל-HX711 מתבצעת באמצעות שני קווים:

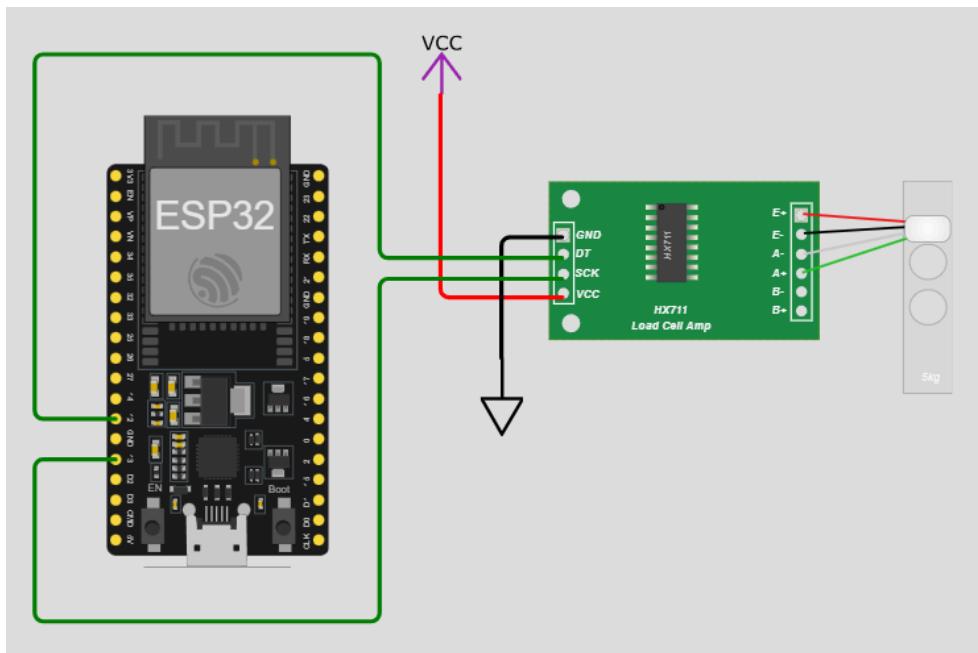
קו שעון - PD_SCK

קו נתונים - DOUT

עקרון העבודה של ערוץ התקשרות:

1. כאשר הנתונים מוכנים, הקו DOUT יורד לרמה נמוכה
2. הבקר שולח פולסים בקו PD_SCK כדי לקרוא את הנתונים
3. אחרי 24 פולסים מתקבל ערך המדידה
4. פולסים נוספים קובעים את ההגבר של הרכיב

להלן חיבור החישון לבקר ESP32:



להלן דוגמת קוד ב-*MicroPython* המציג את הימוש הבו-ו':

```

from machine import Pin
from time import sleep_us, sleep_ms

class HX711:
    def __init__(self, pd_sck, dout, gain=128):
        self.pd_sck = Pin(pd_sck, Pin.OUT)
        self.dout = Pin(dout, Pin.IN)
        self.gain = gain

    def is_ready(self):
        return self.dout.value() == 0

    def read(self):
        # נחכה שהנתונים יהיו מוכנים #
        while not self.is_ready():
            pass

        # נקרא 24 ביטים של נתונים #
        data = 0
        for _ in range(24):
            self.pd_sck.value(1)
            sleep_us(1)
            data = (data << 1) | self.dout.value()
            self.pd_sck.value(0)
            sleep_us(1)

        # נשלח פולסים נוספים בהתאם להגבר הרצוי #
        for _ in range(self.gain == 128 and 1 or self.gain == 64 and 3 or 2):
            self.pd_sck.value(1)
            sleep_us(1)
            self.pd_sck.value(0)

```

```

sleep_us(1)

# נטפל במספר שלילי
if data & 0x800000:
    data = data - 0x1000000

return data

def read_average(self, times=3):
    sum = 0
    for _ in range(times):
        sum += self.read()
        sleep_ms(100)
    return sum / times

# דוגמת שימוש
hx = HX711(pd_sck=13, dout=12)
value = hx.read_average()
OFFSET = hx.read_average(times=10)
print("OFFSET=", OFFSET)
SCALE = 10583.76/100
print("SCALE=", SCALE)
while True:
    value = hx.read_average()
    print((OFFSET-value)/SCALE)
    sleep_ms(1000)

```

נקבל את הפלט הבא:

```

Shell x
>>> %Run -c $EDITOR_CONTENT
MPY: soft reboot
OFFSET= 134737.7
SCALE= 105.8376
0.4223747
0.04133692
32.10609
99.44518
99.56166
99.58691
99.51442

```

כדי לקבל מדידות משקל מדויקות, יש צורך בכיוול החישון. תהליכי הכיוול כוללים:

1. מדידת ערך האפס (כשאין משקל)
2. מדידה עם משקל ידוע (לדוגמה 100g)

3. חישוב פקטור המרה בין הערך הדיגיטלי למשקל בק"ג

חישוב לצין שיש להתחשב בגורמים הבאים:

- יציבות מתח ההזנה
- טמפרטורת הסביבה
- רעשים חממיים
- מיקום המשקל על החישון

להלן דוגמת קוד המטפלת בחישון המשקל בצורה יعلاה יותר:

יש לשמור את הקובץ הבא בשם `uq711.py` ברכיב

```
from machine import Pin, enable_irq, disable_irq, idle

class HX711:
    def __init__(self, dout, pd_sck, gain=128):

        self.pSCK = Pin(pd_sck, mode=Pin.OUT)
        self.pOUT = Pin(dout, mode=Pin.IN, pull=Pin.PULL_DOWN)
        self.pSCK.value(False)

        self.GAIN = 0
        self.OFFSET = 0
        self.SCALE = 1

        self.time_constant = 0.1
        self.filtered = 0

        self.set_gain(gain);

    def set_gain(self, gain):
        if gain is 128:
            self.GAIN = 1
        elif gain is 64:
            self.GAIN = 3
        elif gain is 32:
            self.GAIN = 2

        self.read()
        self.filtered = self.read()
        print('Gain & initial value set')

    def is_ready(self):
        return self.pOUT() == 0

    def read(self):
        # wait for the device being ready
        while self.pOUT() == 1:
            idle()
```

```

# shift in data, and gain & channel info
result = 0
for j in range(24 + self.GAIN):
    state = disable_irq()
    self.pSCK(True)
    self.pSCK(False)
    enable_irq(state)
    result = (result << 1) | self.pOUT()

# shift back the extra bits
result >>= self.GAIN

# check sign
if result > 0x7fffff:
    result -= 0x1000000

return result

def read_average(self, times=3):
    sum = 0
    for i in range(times):
        sum += self.read()
    return sum / times

def read_lowpass(self):
    self.filtered += self.time_constant * (self.read() - self.filtered)
    return self.filtered

def get_value(self, times=3):
    return self.read_average(times) - self.OFFSET

def get_units(self, times=3):
    return self.get_value(times) / self.SCALE

def Calibrate(self, times=15):
    sum = self.read_average(times)
    self.set_offset(sum)

def set_scale(self, scale):
    self.SCALE = scale

def set_offset(self, offset):
    self.OFFSET = offset

def set_time_constant(self, time_constant = None):
    if time_constant is None:
        return self.time_constant
    elif 0 < time_constant < 1.0:
        self.time_constant = time_constant

def power_down(self):
    self.pSCK.value(False)
    self.pSCK.value(True)

def power_up(self):
    self.pSCK.value(False)

```

להלן קובץ להדגמת המחלקה hx711.py

```
from hx711 import HX711
from machine import reset
import time

hx = HX711(12, 13)
hx Calibrate()           #Calibrate the weight to zero
hx.set_scale(-105.4)     #Converts data to weight in grams

try:
    while True:
        val = hx.get_units(10)
        print(val)
        time.sleep_ms(500)

except KeyboardInterrupt:
    print('Ctrl-C pressed...exiting')
    reset()
```

נקבל את הפלט הבא:

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Gain & initial value set
-0.4044117
0.03958136
103.68
100.1078
100.0528
100.6079
100.2634
100.1772
100.1999
Ctrl-C pressed...exiting
```

משימה 24 - שירות העברת מסרים מבוסס Telegram

קישורים:

<https://core.telegram.org/bots/tutorial#obtain-your-bot-token>

סביר תחילה מה זה Bot Token ו- בקשר ל-Telegram

ב-Telegram, בוט הוא יישום מתוכנת שמבצע משימות באופן אוטומטי. בוטים נועדו לתקשר עם משתמשים באמצעות הודעות, להגיב לפקודות, לספק מידע, לנוהל פעולות, ואפיו לשלב שירותים חיצוניים. ניתן להשתמש בבוטים למטרות שונות, כגון צ'אט אוטומטי, ניהול קהילות, יצירהTZCOROT, אינטגרציה עם מערכות חיצונית.

שירות Telegram מספק לנו בוטים ללא עלות.

Bot Token הוא מפתח גישה ייחודי שנמצא ב-@BotFather, הבוט האחראי לניהול והגדרה של בוטים ב-Telegram. כאשר יוצרים בוט חדש באמצעות @BotFather, הוא מנפיק את הטוקן הזה כדי לאפשר למפתחים לתקשר עם API Telegram. הטוקן משמש לאימות זהה של הבוט, והוא נדרש כדי לבצע פעולות כמו שליחת הודעות, קבלת פקודות ותפעול תהליכי. חשוב לשמור על הטוקן בסודיות כדי למנוע גישה לא מורשית.

שלב 1: קבלת Bot Token מאתר Telegram או דרך האפליקציה בטלפון הנייד

גישה ל-[@BotFather](https://t.me/BotFather).

יש להכנס לאפליקציה ולחפש את @BotFather, ולהתחליל אליו שיחה. יש להקליד את הפקודה /newbot ולשלוח אותה ל-BotFather. הבוט ידריך אתכם בתהליך יצירת הבוט, כולל בחירת שם לבוט (שם תצוגה) ושם משתמש ייחודי, שצריך להסתTEM ב-bot (לדוגמא: "MyESP32Bot"). לאחר השלמת התהליך, יספק לנו Token Bot Token, מפתח גישה ייחודי שמשמש לחיבור הבוט ל-API Telegram. עלייכם לוודא שה-Token נשמר במקום בטוח. ה-Token משמש לזיהוי ואימות של הבוט, וכל מי שייחסק בו יוכל לשולוט בו.



לאחר שיחה קצרה עם הבוט קיבלנו Token שיראה בערך כך:

4839574812:AAFD39kkdpWt3ywRZergyOLMaJhac60qc



להלן קוד המחלקה המתאפשרת בפנויות ל-Telegram. יש לשמר את הקובץ בשם telegram.py בזיכרון הפנימי של ה-ESP32:

```
import time
import gc
import urequests

class ubot:

    def __init__(self, token, offset=0):
        self.url = 'https://api.telegram.org/bot' + token
        self.commands = {}
        self.default_handler = None
        self.message_offset = offset
        self.sleep_btwn_updates = 3

        messages = self.read_messages()
        if messages:
            if self.message_offset==0:
                self.message_offset = messages[-1]['update_id']
            else:
                for message in messages:
                    if message['update_id'] >= self.message_offset:
                        self.message_offset = message['update_id']
                        break

    def send(self, chat_id, text):
        data = {'chat_id': chat_id, 'text': text}
        try:
            headers = {'Content-type': 'application/json', 'Accept': 'text/plain'}
            response = urequests.post(self.url + '/sendMessage', json=data, headers=headers)
            response.close()
            return True
        except:
            return False

    def read_messages(self):
        messages = []
        offset = self.message_offset
        while True:
            params = {'offset': offset, 'timeout': 10}
            response = urequests.get(self.url + '/getUpdates', params=params)
            if response.status_code == 200:
                data = response.json()
                if len(data) > 0:
                    for message in data:
                        if message['update_id'] > offset:
                            messages.append(message)
                            offset = message['update_id']
                else:
                    break
            else:
                break
        return messages
```

```

        return False

def read_messages(self):
    result = []
    self.query_updates = {
        'offset': self.message_offset + 1,
        'limit': 1,
        'timeout': 30,
        'allowed_updates': ['message']}}

    try:
        update_messages = urequests.post(self.url + '/getUpdates',
json=self.query_updates).json()
        if 'result' in update_messages:
            for item in update_messages['result']:
                result.append(item)
    return result
    except (ValueError):
        return None
    except (OSError):
        print("OSError: request timed out")
        return None

def listen(self):
    while True:
        self.read_once()
        time.sleep(self.sleep_btwn_updates)
        gc.collect()

def read_once(self):
    messages = self.read_messages()
    if messages:
        if self.message_offset==0:
            self.message_offset = messages[-1]['update_id']
            self.message_handler(messages[-1])
        else:
            for message in messages:
                if message['update_id'] >= self.message_offset:
                    self.message_offset = message['update_id']
                    self.message_handler(message)
                    break

    def register(self, command, handler):
        self.commands[command] = handler

    def set_default_handler(self, handler):
        self.default_handler = handler

    def set_sleep_btwn_updates(self, sleep_time):
        self.sleep_btwn_updates = sleep_time

    def message_handler(self, message):
        if 'text' in message['message']:
            parts = message['message']['text'].split(' ')

```

```

        if parts[0] in self.commands:
            self.commands[parts[0]](message)
        else:
            if self.default_handler:
                self.default_handler(message)

```

להלן קוד העושה שימוש במחלקה botn בקובץ utelegram.py לצורך התחברות לערוץ שלכם בטלגרם תוך שימוש ב- Token שקיבלתם:

```

import utelegram

def get_message(message):
    print(message)
    bot.send(message['message']['chat']['id'],
    message['message']['text'].upper())

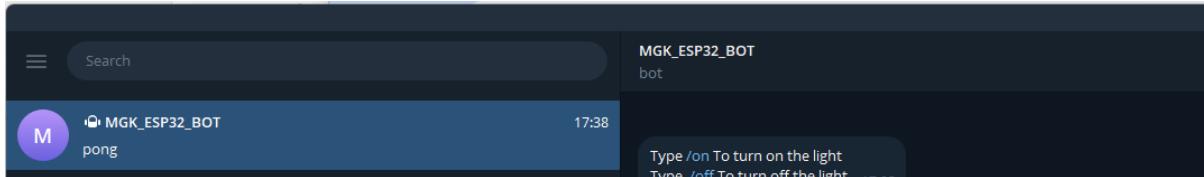
def reply_ping(message):
    print(message)
    bot.send(message['message']['chat']['id'], 'pong')

bot = utelegram.ubot('7_____3:A_____A')
bot.register('/ping', reply_ping)
bot.set_default_handler(get_message)

print('BOT LISTENING')
bot.listen()

```

מניתוך הקוד ניתן לראות כיצד קוד זה עושה שימוש ב- Token כדי להתחבר לערוץ שיצרתם תוך כדי הכתיבה עם `@BotFather`. במקרה של יצرتם בעזרת `@BotFather` בשם `MGK_ESP32_BOT`



ניתן בעזרה `@BotFather` לשנות את שם הערוץ, לקבוע תמונה לוגו, להגדיר תיאור לערוץ ובKİיזור לטפל בכל מה שקשרו לערוץ שיצרתם.

הסבר הקוד

יכרנו את שני פועלות הבאות

```

def get_message(message):
    print(message)

```

```

bot.send(message['message']['chat']['id'],
message['message']['text'].upper())

def reply_ping(message):
    print(message)
    bot.send(message['message']['chat']['id'], 'pong')

```

פעולה ראשונה בשם get_message מופעלת בכל פעם שמתאפשר טקסט אחד המנויים הרשומים בעברוץ. הפעולה זו נרשמת בעזרת הקוד הבא:

```
bot.set_default_handler(get_message)
```

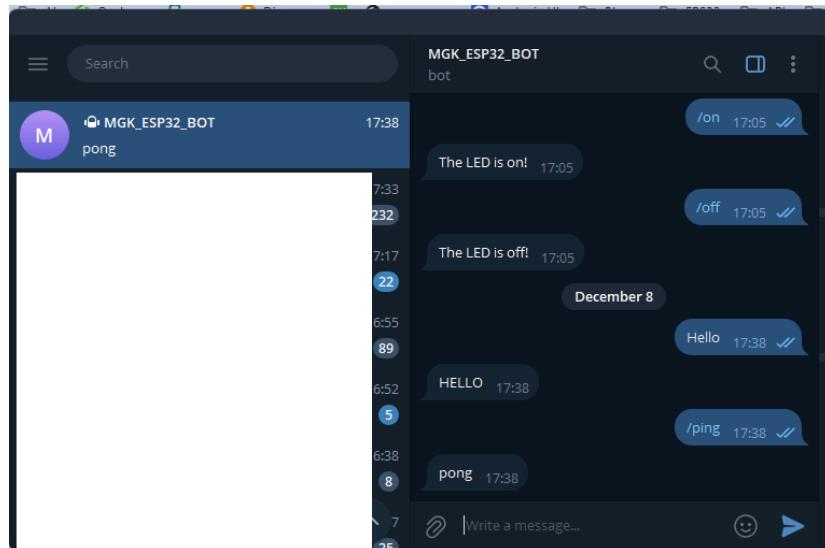
פעולה שנייה בשם reply_ping מופעלת בכל פעם שאחד המנויים הרשומים בעברוץ שלוח את הטקסט "/ping". הפעולה זו נרשמת בעזרת הקוד הבא:

```
bot.register('/ping', reply_ping)
```

שימוש בשלב שהפעולות שלוחת חזרה והודעה ללקוק על ידי הפעולה הבא:

```
bot.send(message['message']['chat']['id'], 'pong')
```

לסיום נקבל את מסך הפלט הבא בצד הלקוק:



בצד היישום שרך על ה-ESP32 נקבל את הפלט הבא:

```

Shell < 
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Already connected
(192.168.202.92, '255.255.255.0', '192.168.202.162', '192.168.202.162')
BOT LISTENING
{'update_id': [REDACTED], 'message': {'message_id': 99, 'from': {'is_bot': False, 'last_name': 'Rman', 'id': [REDACTED], 'first_name': 'Gadi'}, 'text': 'Hello', 'date': [REDACTED]}, 'chat': {'type': 'private', 'first_name': 'Gadi'}})
{'update_id': [REDACTED], 'message': {'message_id': 91, 'from': {'is_bot': False, 'last_name': 'Herman', 'id': [REDACTED], 'first_name': 'Gadi'}, 'text': '/ping', 'date': [REDACTED]}, 'entities': {'bot_command': []}, 'chat': {'id': [REDACTED], 'last_name': 'Herman', 'type': 'private'}},

```

להלן דוגמת קוד נוספת:

```

import utime
from machine import Pin
pin_led = Pin(2, mode=Pin.OUT)

def get_message(message):
    print('update_id: ', message['update_id'], '\n')
    print('-----')
    print('message/message_id: ', message['message']['message_id'], '\n')
    print('message/from: ', message['message']['from'], '\n')
    print('message/text: ', message['message']['text'], '\n')
    print('message/date: ', message['message']['date'], '\n')
    print('message/chat: ', message['message']['chat'], '\n')
    print('-----')
    print('message/chat/id: ', message['message']['chat']['id'], '\n')
    print('message/chat/last_name: ', message['message']['chat']['last_name'], '\n')
    print('message/chat/type: ', message['message']['chat']['type'], '\n')
    print('message/chat/first_name: ', message['message']['chat']['first_name'], '\n')
    print('-----')
    bot.send(message['message']['chat']['id'], 'Type /on To turn on the light\nType /off To turn off the light')

def led_on(message):
    print(message)
    pin_led.on()
    bot.send(message['message']['chat']['id'], 'The LED is on!')

def led_off(message):
    id = message['message']['chat']['id']

```

```

print(id)
pin_led.off()
bot.send(id, 'The LED is off!')

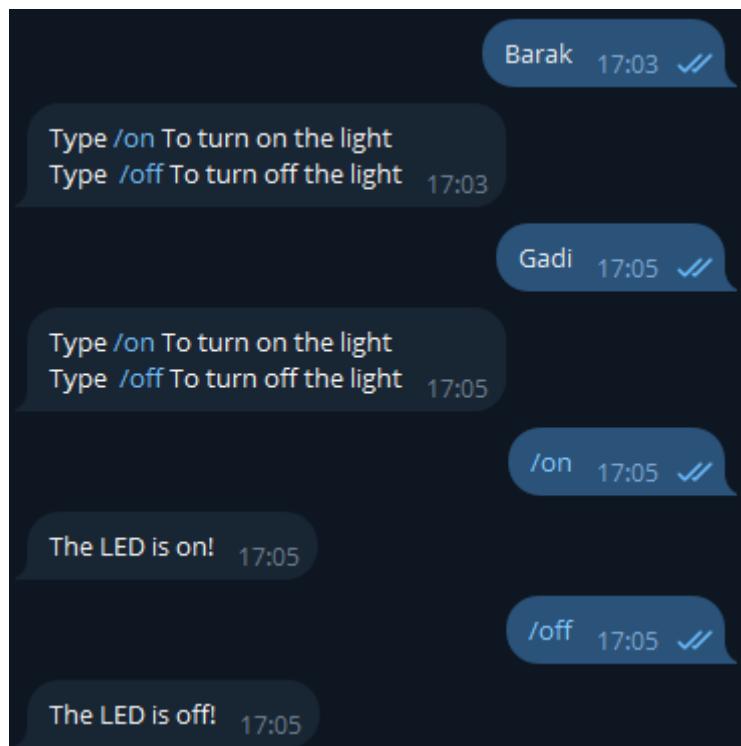
bot = utelegram.ubot('7_____3:A_____A')
bot.register('/on', led_on)
bot.register('/off', led_off)
bot.set_default_handler(get_message)

print('BOT LISTENING')
bot.listen()

```

קוד זה מציג כיצד ניתן להפעיל נורת LED תוך שימוש בהודעות טקסט באופן הבא:

- הדלקת הנורת על ידי שליחת הודעה: "/on"
- כיבוי הנורת על ידי שליחת הודעה: "/off"
- כל שליחת הודעה שונה מ-"on" או "off" תעביר ללקוט טקסט הסבר מה לכתוב.



מシימת 25 - חיישן טביעה אצבע - DY50

קישורים:

<https://github.com/chrisb2/micropython-fingerprint>

<https://github.com/bastianraschke/pyfingerprint/tree/Development/src/files/examples>

חיישן טביעה אצבע מדגם DY50 הוא רכיב לזיהוי ביומטרי המשמש לאימות וזיהוי זהות באמצעות טביעות אצבע. להלן הסבר מפורט על מאפייניו ועקרונות פועלתו:

עיקנון פעולה וטכנולוגית דגימת טביעה אצבע

החיישן משתמש בטכנולוגיית דגימה אופטית לצורך צילום וניתוח של משטח העור באצבע. תהליך הדגימה מתבצע באופן הבא:

1. קלילת התמונה:

- החיישן מכיל חיישן אופטי המורכב ממטריצת פיקסליםDKIKה
- בעת הנחת האצבע, אור LED פוגע במשטח העור
- האזוריים הבולטים והשകעים בטביעה האצבע יוצרים החזרי אוור שונים
- חיישן קולט את ההבדלים בהחזרי האור ויוצר תמונה דיגיטלית

2. עיבוד התמונה:

התמונה הדיגיטלית עוברת עיבוד דיגיטלי על ידי מיפוי טביעה האצבע ואז ייצור "תבנית" מתמטית ייחודית המתארת את מאפייני טביעה האצבע.

3. אחסון וזיהוי:

- התבנית נשמרת בזכרון פנימי של החיישן
- כל תבנית توוספת C-512 בתים
- ניתן לאחסן עד כ-200 תבניות שונות
- בעת השוואה, מתבצע חישוב סטטיסטי של מידת ההתאמה בין התבניות לטביעה האצבע

מאפיינים טכניים עיקריים

- מתח עבודה: 3.3-5 וולט
- זרם צריכה: 100-500 מיליאΜפר
- תקשורת: UART סיבית סטנדרטית
- קצב תקשורת: 57600 ביט/שניה
- נפח זיכרון: 32 קילובייט
- מספר תבניות מרבי: עד 200 תבניות
- רזולוציית דגימה: DPI 500
- דיוק זיהוי: כ-99.9%
- מהירות זיהוי: פחות מ-1 שניה

תקשורת עם ESP32

- RX מהחיישן מחובר ל- TX של ESP32
- TX מהחיישן מחובר ל- RX של ESP32
- חיבור משותף של (GND)

- מתח 3.3V מסווק מיציאת מתח של ESP32

פרוטוקול תקשורת

הפרוטוקול מבוסס על חבילות נתונים כאשר כל נתונים בגודל Byte כמפורט בתקשורת UART:

כל חבילה תקשורת כוללת:

- סימן התחלה קבוע (0xEF, 0x01)
- כתובות התקן (4 בתים)
- אורך הפקודה
- סוג הפקודה
- פרמטרים נוספים
- Checksum

רצף פעולות אופייני:

1. שליחת פקודת ליכידת תמונה
2. אימוח איות התמונה
3. ייצור תבנית מתמטית
4. שמירה/השווה עם התבניות קיימות

ניטוח פרוטוקול תקשורת DY50

EF - קוד התחלה קבוע, מגדר את תחילת חבילת התקשורת
0x01 - המישר קוד התחלה

0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF - כתובות ברירתי מחדל של החישון

0x01 - קוד הבקשה, מצין סוג פעולה או בקשה לחישון

0x00 - קוד המציין את אורך הפקודה הנשלחת

0x04 0x03 0x07 0x01 - קוד נוסף המכיל פרטיים של הפקודה

0x0F 0x02 0x01 0x13 0x01 0x02 0x01 - קוד פקודה המגדיר את סוג הפעולה כמו: סריקה, ייצור תבנית, וצדומה..

Checksum - מאפשר זיהוי שגיאות בהעברת הנתונים

לדוגמה:



תקשורת זו מדגימה את אתחול תקשורת בין בקר ESP32 לבין חישון DY50.

קוד תגובה עיקריים

0x00: הצלחה

0x01: שגיאה

0x02: קבצים לא תקינים

0x03: תקשורת לא תקינה

הערות נוספות:

כל בית מועבר כערך 8 ביט

סדר העברת נתונים: מהבית הפחות משמעותית לבית המשמעותי ביותר

Checksum מחושב על ידי סכימת הבטים המשודרים

להלן קישור להורדת קוד מחלקה המממשת את הUART ופונקציית הבדיקה של החישון. יש להוריד את הקוד דרך הקישור הבא ולשמור את הקובץ בשם pyfingerprint.py בדיסק עצמו.

https://github.com/GadiHerman/ESP32_MicroPython_AllBookFiles/blob/main/25_Fingerprint/pyfingerprint.py

התחברות ראשונית ובדיקה סיסמה:

```
from machine import UART
import sys
from pyfingerprint import PyFingerprint

try:
    f = PyFingerprint(UART(2, 57600), 0xFFFFFFFF, 0x00000000)
    if (f.verifyPassword()):
        print('Connected successfully and the password is correct.')
        print('Currently used templates: ' + str(f.getTemplateCount()))
    else:
        print('The fingerprint sensor password is wrong.')
        sys.exit()
except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Error: ', str(e))
    sys.exit()
```

נקבל את הפלט הבא:

```
Shell >>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Connected successfully and the password is correct.
Currently used templates: 5/150
```

במידה וקיבלהת את ההודעה הנ"ל סימן שיש לכם גישה לחישון ונitin להתקדם לשלב הבא שהוא שמירת טביעה אצבע חדשה בחישון.

להלן קוד השומר טביעה אצבע חדשה בחישון:

```
import time
import sys
from machine import UART
from pyfingerprint import PyFingerprint
from pyfingerprint import FINGERPRINT_CHARBUFFER1
from pyfingerprint import FINGERPRINT_CHARBUFFER2

try:
    f = PyFingerprint(UART(2, 57600), 0xFFFFFFF, 0x00000000)
    if ( f.verifyPassword() ):
        print('Connected successfully and the password is correct.')

    print('Waiting for finger...')
    while ( f.readImage() == False ):
        pass

    f.convertImage(FINGERPRINT_CHARBUFFER1)

    result = f.searchTemplate()
    positionNumber = result[0]

    if ( positionNumber >= 0 ):
        print('Template already exists at position #' +
str(positionNumber))
        sys.exit()

    print('Remove finger...')
    time.sleep(2)

    print('Waiting for same finger again...')
    while ( f.readImage() == False ):
        pass

    f.convertImage(FINGERPRINT_CHARBUFFER2)
```

```

if ( f.compareCharacteristics() == 0 ):
    raise Exception('Fingers do not match')

f.createTemplate()
positionNumber = f.storeTemplate()
print('Finger enrolled successfully!')
print('New template position #' + str(positionNumber))

except Exception as e:
    print('Error: ', str(e))
    sys.exit()

```

נקבל את הפלט הבא:

```

Shell ×
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Connected successfully and the password in correct.
Waiting for finger...
Remove finger...
Waiting for same finger again...
Finger enrolled successfully!
New template position #7

```

נכתב עכשוו קוד תוכנה הבזק האם אצבע חדשה מותאמת לאחת מטביעות האצבע בחישון:

```

import time
import sys
from machine import UART
from pyfingerprint import PyFingerprint
from pyfingerprint import FINGERPRINT_CHARBUFFER1

try:
    f = PyFingerprint(UART(2, 57600), 0xFFFFFFFF, 0x00000000)
    if ( f.verifyPassword() ):
        print('Connected successfully and the password in correct.')

    print('Waiting for finger...')
    while ( f.readImage() == False ):
        pass

    f.convertImage(FINGERPRINT_CHARBUFFER1)
    result = f.searchTemplate()
    positionNumber = result[0]
    accuracyScore = result[1]

    if ( positionNumber == -1 ):
        print('No match!')

```

```

        sys.exit()
else:
    print('Found template at position #' + str(positionNumber))
    print('The accuracy score is: ' + str(accuracyScore))

except Exception as e:
    print('Error: ', str(e))
    sys.exit()

```

נקבל את הפלט הבא (כאשר יש זיהוי):

Shell ×

>>> %Run -c \$EDITOR_CONTENT

```

MPY: soft reboot
Connected successfully and the password is correct.
Waiting for finger...
Found template at position #2
The accuracy score is: 165

```

נקבל את הפלט הבא (כאשר אין זיהוי):

Shell ×

>>> %Run -c \$EDITOR_CONTENT

```

MPY: soft reboot
Connected successfully and the password is correct.
Waiting for finger...
No match!
MPY: soft reboot

```

מחיקת טביעות אצבע:

```

else:
    print('Found template at position #' + str(positionNumber))
    print('The accuracy score is: ' + str(accuracyScore))

except Exception as e:
    print('Error: ', str(e))
    sys.exit()

```

נקבל את הפלט הבא:

Shell ×

```
MPY: soft reboot
Connected successfully and the password is correct.
Used templates: 6/150
Fingerprint at position #0 is in used.
Fingerprint at position #3 is in used.
Fingerprint at position #4 is in used.
Fingerprint at position #5 is in used.
Fingerprint at position #6 is in used.
Fingerprint at position #7 is in used.
Enter the Fingerprint position you want to delete: 0
Template deleted!
Used templates: 5/150
Fingerprint at position #3 is in used.
Fingerprint at position #4 is in used.
Fingerprint at position #5 is in used.
Fingerprint at position #6 is in used.
Fingerprint at position #7 is in used.
```

משימה 26 - מערכת הקבצים של בקר ESP32

קישורים:

<https://docs.micropython.org/en/latest/library/os.html>

<https://docs.micropython.org/en/latest/reference/filesystem.html>

<https://docs.micropython.org/en/latest/library/hashlib.html>

תומך במספר מערכות לניהול קבצים: ESP32

- SPIFFS _ Serial Peripheral Interface Flash File System
- فلاש
- LittleFS
- - במקורה של שימוש בכרטיס SD חיצוני

היא מערכת קבצים המיעודת למיכנירים מבוססי فلاש, והיא הרבה יותר עמידה בפני קריסה של מערכת הקבצים. لكن זו מערכת הקבצים העדיפה לשימוש במערכת הקבצים של ESP32.

כדי לבדוק איזו מערכת קבצים יש כרגע בברker ניתן להריץ את הקוד הבא:

```
import sys, struct, esp

def pr(addr, label, data):
    if type(data) == bytes or type(data) == bytearray:
        data = ':'.join('%02x' % b for b in data)
    print('%04x %-32s %s' % (addr, label, data))

def decode_bootsec_fat(b):
    print('FAT filesystem')
    pr(0, 'jump', b[0:3])
    pr(3, 'OEM name', str(b[3:11], 'ascii'))
    pr(11, 'sector size (bytes)', struct.unpack_from('<H', b, 11)[0])
    pr(13, 'cluster size (sectors)', struct.unpack_from('<B', b, 13)[0])
    pr(14, 'reserved area (sectors)', struct.unpack_from('<H', b, 14)[0])
    pr(16, 'number of FATs', struct.unpack_from('<B', b, 16)[0])
    pr(17, 'size of root dir area', struct.unpack_from('<H', b, 17)[0])
    pr(19, 'volume size (sectors)', struct.unpack_from('<H', b, 19)[0])
    pr(21, 'media descriptor', hex(struct.unpack_from('<B', b, 21)[0]))
    pr(22, 'FAT size (sectors)', struct.unpack_from('<H', b, 22)[0])
    pr(24, 'track size (sectors)', struct.unpack_from('<H', b, 24)[0])
    pr(26, 'number of heads', struct.unpack_from('<H', b, 26)[0])
    pr(28, 'volume offset (sectors)', struct.unpack_from('<L', b, 28)[0])
    pr(32, 'volume size (32-bit) (sectors)', struct.unpack_from('<I', b, 32)[0])
    pr(36, 'physical drive number', struct.unpack_from('<B', b, 36)[0])
    pr(37, 'error flag', hex(struct.unpack_from('<B', b, 37)[0]))
    pr(38, 'extended boot signature', hex(struct.unpack_from('<B', b, 38)[0]))
    pr(39, 'volume serial number', hex(struct.unpack_from('<L', b, 39)[0]))
    pr(43, 'volume label', str(b[43:51], 'ascii'))
    pr(54, 'filesystem type', str(b[54:62], 'ascii'))
    pr(510, 'signature', hex(struct.unpack_from('<H', b, 510)[0]))

def decode_bootsec_lfs1(b):
    print("Littlefs v1 filesystem")
    pr(24, "type", struct.unpack_from("<I", b, 24)[0])
```

```

pr(25, "elen", struct.unpack_from("<I", b, 25)[0])
pr(26, "alen", struct.unpack_from("<I", b, 26)[0])
pr(27, "nlen", struct.unpack_from("<I", b, 27)[0])
pr(28, "block_size", struct.unpack_from("<I", b, 28)[0])
pr(32, "block_count", struct.unpack_from("<I", b, 32)[0])
pr(36, "version", hex(struct.unpack_from("<I", b, 36)[0]))
pr(40, "magic", str(b[40:40 + 8], 'ascii'))

def decode_bootsec_lfs2(b):
    print("Littlefs v2 filesystem")
    pr(8, "magic", str(b[8:8 + 8], "ascii"))
    pr(20, "version", hex(struct.unpack_from("<I", b, 20)[0]))
    pr(24, "block_size", struct.unpack_from("<I", b, 24)[0])
    pr(28, "block_count", struct.unpack_from("<I", b, 28)[0])
    pr(32, "name_max", struct.unpack_from("<I", b, 32)[0])
    pr(36, "file_max", struct.unpack_from("<I", b, 36)[0])
    pr(40, "attr_max", struct.unpack_from("<I", b, 40)[0])

def decode_bootsec(b):
    if b[40:48] == b"littlefs":
        decode_bootsec_lfs1(b)
    elif b[8:16] == b"littlefs":
        decode_bootsec_lfs2(b)
    else:
        decode_bootsec_fat(b)

def main():
    bootsec = bytearray(512)
    esp.flash_read(esp.flash_user_start(), bootsec)
    decode_bootsec(bootsec)

main()

```

נקבל את הפלט הבא:

```

Shell >
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Littlefs v2 filesystem
0008  magic           littlefs
0014  version         0x20001
0018  block_size       4096
001c  block_count      512
0020  name_max         255
0024  file_max         2147483647
0028  attr_max         1022

>>>

```

כדי לשנות את מערכת ניהול הקבצים של הAKER שלכם ניתן להשתמש באחד משני הקודים הבאים.

זהירות !!! שני הקודים הבאים מוחקם את כל הקבצים השמורים בזיכרון של הAKER

כדי להתקין מערכת לניהול קבצים מסוג FAT כולל אתחול הזיכרון של הAKER:

```
import vfs
vfs.umount('/')
vfs.VfsFat.mkfs(bdev)
vfs.mount(bdev, '/')
```

כדי להתקין מערכת לניהול קבצים מסוג LittleFS כולל אתחול הזיכרון של הAKER:

```
import vfs
vfs.umount('/')
vfs.VfsLfs2.mkfs(bdev)
vfs.mount(bdev, '/')
```

לאחר קביעת המערכת שמנוהלת את הקבצים בAKER שלכם נדגים בשלב זה כיצד ניתן לנוהל את מערכ הקבצים תוך שימוש בקוד.

להלן קוד המציג מידע על הקבצים השמורים בזיכרון הAKER:

```
import os

def get_fs_info():
    try:
        stats = os.statvfs('/')
        total_space = stats[0] * stats[2]  # block_size * total_blocks
        free_space = stats[0] * stats[3]   # block_size * free_blocks
        return {
            'total_space': total_space,
            'free_space': free_space
        }
    except Exception as e:
        return f":שגיאה בקבלת מידע על מערכת הקבצים {str(e)}"

def list_files():
    try:
        return os.listdir()
    except Exception as e:
        return f":שגיאה בהצגת קבצים {str(e)}"

info = get_fs_info()
print(":\n{:>15} {}".format("מיצט", "מערכת הקבצים"))
print(f"\t{:>15} {:>15} bytes".format("נפח כולל", info['total_space']))
print(f"\t{:>15} {:>15} bytes".format("נפח פנוי", info['free_space']))

print("\nרשימת קבצים:")
files = list_files()
for f in files:
```

```
print(f"- {f}")
```

נקבל את הפלט הבא:

```
Shell >>> %Run -c $EDITOR_CONTENT

MPY: soft reboot

: מידע על מערכת הקבצים
2097152 bytes נפח כולל: bytes
2076672 bytes נפח פנוי: bytes

: רשימת קבצים
- ReadJSON.py
- data.json
- rtc1.py
- time.json

>>>
```

הקוד שלහן מציג תפריט אינטראקטיבי המאפשר:

- הצגת מידע על מערכת הקבצים (נפח כולל ופנוי)
- הצגת רשימת הקבצים הקיימים
- ייצירת קובץ חדש
- קריאה מקובץ
- מחיקת קובץ

הקוד כולל טיפול בשגיאות בכל הפונקציות

שימוש ב-gc.collect לניקוי זיכרון שאינו בשימוש

תמייה בעברית מלאה במסמך המשמש

```
import os
import gc

def get_fs_info():
    """מחזיר מידע על מערכת הקבצים"""
    try:
        # חישוב נפח כולל וזמן
        stats = os.statvfs('/')
        total_space = stats[0] * stats[2]  # block_size * total_blocks
        free_space = stats[0] * stats[3]   # block_size * free_blocks
```

```

        return {
            'total_space': total_space,
            'free_space': free_space
        }
    except Exception as e:
        return f":שגיאה בקבלה מידע על מערכת הקבצים" {str(e)}"

def list_files():
    """מציג את כל הקבצים במערכת"""
    try:
        return os.listdir()
    except Exception as e:
        return f":שגיאה בהצגת קבצים" {str(e)}"

def create_file(filename, content):
    """يُוצר كوبץ جديد مع محتوى مسحوب"""
    try:
        with open(filename, 'w') as f:
            f.write(content)
        return f"النفاذ بـ {filename} النجاح"
    except Exception as e:
        return f":שגיאה في إنشاء الكوبـ" {str(e)}"

def read_file(filename):
    """كوراء محتوى كوبـ"""
    try:
        with open(filename, 'r') as f:
            return f.read()
    except Exception as e:
        return f":שגיאה بـ القراءة من الكوبـ" {str(e)}"

def delete_file(filename):
    """محـكـ كوبـ"""
    try:
        os.remove(filename)
        return f"النـفـاذ بـ {filename} النجاح"
    except Exception as e:
        return f":שגיאה بـ حـلـكـ الكوبـ" {str(e)}"

def print_menu():
    """מציג תפריט אפשרויות"""
    print("==== تـفـريـت مـعـرـقـتـ كـبـزـيم \n")
    print("1. (הציג מידע על מערכת הקבצים")
    print("2. (הציג רשימה קבצים")
    print("3. (צור كوبـ جديد")
    print("4. (كراء مـكـوبـ")
    print("5. (محـكـ كوبـ")
    print("6. (ويـزـيـاهـ")
    print("===== ")

def main_menu():
    """لـولـاتـ التـفـريـت الرـاسـيـ"""
    while True:
        print_menu()

```

```

choice = input("1-6) : (בחר אפשרות (1-6)

if choice == '1':
    info = get_fs_info()
    print("מיצג על מערכת הקבצים")
    print(f"נפח כויל {info['total_space']} bytes")
    print(f"נפח פנוי {info['free_space']} bytes")

elif choice == '2':
    print("רשימת קבצים")
    files = list_files()
    for f in files:
        print(f"- {f}")

elif choice == '3':
    filename = input("שם הקובץ:")
    content = input("תוכן הקובץ:")
    print(create_file(filename, content))

elif choice == '4':
    filename = input("שם הקובץ לקריאה:")
    print("תוכן הקובץ:")
    print(read_file(filename))

elif choice == '5':
    filename = input("שם הקובץ למחיקה:")
    print(delete_file(filename))

elif choice == '6':
    print("!להתראות")
    break

else:
    print("אפשרות לא חוקית, נסה שוב")

print("...להמשך Enter")
input()
gc.collect() # ניקוי זיכרון לא בשימוש

if __name__ == '__main__':
    main_menu()

```

משימה 27 - תבנית לפרויקט המשלב מערכת לניהול משתמשים (הוּא שימוש בהצפנה!!!)

קישורים:

<https://docs.micropython.org/en/latest/library/hashlib.html>

לא פעם יצא לי להנחות פרויקט הכלול דרישת ניהול משתמשים, לדוגמה בית חכם, שבו יש מספר משתמשים שלכל אחד מהם יש סיסמה ייחודית. במצב זה אני רואה כיצד הלומד בונה קוד באופן הבא:

```
username = input("הכנס שם משתמש חדש")
password = input("הכנס סיסמה")
if username=='admin' and password=='1234':
    print("התחברת בהצלחה!")
else:
    print("שם משתמש או סיסמה שגויה")
```

קוד זה תקין מבחינה תכנית אבל הוא אסון מבחינה יישומית, בגלל שכל שינוי סיסמה דורש שינוי קוד. כמו כן כל אחד בעל ידע בסיסי שוראה את קוד התוכנית יכול להבין מהו השם המשתמש ומה הסיסמה!!!!

בפועלות זו נפתרת את שתי הבעיה שבקוד זה על ידי כך שבשלב הראשון נתמודד עם הבעה ששני סיסמה דורש שינוי קוד. נעשה זאת על ידי יצירת קובץ טקסט חיצוני הכלול רישימה של שמות וסיסמות. באופן זה ניתן יהה לשמר סיסמות מחוץ למועד התוכנה.

בשלב השני נתמודד גם את הבעיה השנייה שבה כל אחד בעל ידע בסיסי שוראה את תוכן הקובץ יכול להבין מה שם המשתמש ומה הסיסמה!!!! נעשה זאת על ידי הצפנה הסיסמה תוך שימוש באלגוריתם SHA256 SHA256 הוא אלגוריתם הצפנה מודרני (מסדרת SHA2). הוא מתאים למטרות אבטחה קריפטוגרפית. הצפנה זו זמינה בספרייה ייחודית בליתת MicroPython .

נתחיל בדוגמה ראשונה השומרת רשימה של משתמשים בקובץ טקסט חיצוני בזיכרון הAKER. קוד זה מאפשר ליצור משתמשים חדשים, להציג רשימה של משתמשים, לעדכן סיסמה של משתמש או למחוק משתמש.

בפעם הראשונה שהתוכנה מופעלת נוצר הקובץ שנשמר בו שם משתמש admin עם סיסמה admin. לאחר ההתחברות הראשונה ניתן להחליף את הסיסמה של ה- admin לסיסמה אחרת.

להלן הקוד:

```
import json
import os

class UserManager:
    def __init__(self, filename="users.txt"):
        self.filename = filename
        self._ensure_file_exists()

    def _ensure_file_exists(self):
        """ণונזר קובץ משתמשים אם לא קיים"""
        if not self.filename in os.listdir():
            with open(self.filename, 'w') as f:
                json.dump({}, f)

    def _read_users(self):
        """קוראת רשימת המשתמשים מהקובץ"""
        with open(self.filename, 'r') as f:
            return json.load(f)
```

```

def _save_users(self, users):
    """שומר את רשימת המשתמשים לקובץ"""
    with open(self.filename, 'w') as f:
        json.dump(users, f)

def verify_login(self, username, password):
    """אמת פרטי התחברות"""
    users = self._read_users()
    return username in users and users[username] == password

def add_user(self, username, password):
    """מוסיף משתמש חדש"""
    users = self._read_users()
    if username in users:
        return False, "שם המשתמש כבר קיים"
    users[username] = password
    self._save_users(users)
    return True, "המשתמש נוסף בהצלחה"

def delete_user(self, username):
    """מחוקק משתמש"""
    users = self._read_users()
    if username not in users:
        return False, "המשתמש לא קיים"
    del users[username]
    self._save_users(users)
    return True, "המשתמש נמחק בהצלחה"

def update_password(self, username, new_password):
    """מעדכן סיסמה למשתמש"""
    users = self._read_users()
    if username not in users:
        return False, "המשתמש לא קיים"
    users[username] = new_password
    self._save_users(users)
    return True, "הסיסמה עודכנה בהצלחה"

def list_users(self):
    """מחזיר רשימת משתמשים"""
    return list(self._read_users().keys())

def admin_menu(user_manager):
    """תפריט ניהול משתמשים"""
    while True:
        print("\n==== תפריט ניהול משתמשים ===")
        print("1. הוסף משתמש חדש")
        print("2. מחוקק משתמש")
        print("3. עדכן סיסמה")
        print("4. הצג רשימת משתמשים")
        print("5. הtentek")

        choice = input("\n1-5) : (בחר אפשרות ) ")

```

```

if choice == "1":
    username = input("הכנס שם משתמש חדש:")
    password = input("הכנס סיסמה:")
    success, message = user_manager.add_user(username, password)
    print(message)

elif choice == "2":
    username = input("הכנס שם משתמש למחיקה:")
    success, message = user_manager.delete_user(username)
    print(message)

elif choice == "3":
    username = input("הכנס שם משתמש:")
    new_password = input("הכנס סיסמה חדשה:")
    success, message = user_manager.update_password(username, new_password)
    print(message)

elif choice == "4":
    users = user_manager.list_users()
    print("\nרשימת משתמשים:")
    for user in users:
        print(f"- {user}")

elif choice == "5":
    print("להתראות!")
    break

else:
    print("אפשרות לא חוקית")

input("\nEnter לחץ...")

def main():
    user_manager = UserManager()

    אם אין משתמשים, צור משתמש ברירת מחדל #
    if not user_manager.list_users():
        user_manager.add_user("admin", "admin")
        print("נוצר משתמש ברירת מחדל: admin/admin")

    while True:
        print("\n==== מערכת התחברות ===")
        username = input("שם משתמש:")
        password = input("סיסמה:")

        if user_manager.verify_login(username, password):
            print("התחברת בהצלחה!")
            admin_menu(user_manager)
            break
        else:
            print("שם משתמש או סיסמה שגויי\ות")
            retry = input("לנסות שוב? (Y/N): ")
            if retry.lower() != 'y':

```

```

        break

if __name__ == "__main__":
    main()

```

נקבל את הפלט הבא:

```

Shell x

>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
: נוצר משתמש בריית מודול admin/admin

==== מערכות החברות ==
שם: admin
שם: admin
סיסמה: admin

התחברת בהצלחה!

==== תפריט ניהול משתמשים ==
1. הוסף משתמש חדש
2. מחק משתמש
3. עדכן סיסמה
4. הצג רשימת משתמשים
5. התנתק

(1-5) בחר אפשרות
1: הכנס שם:gadi
2: הכנס סיסמה:1234
המשתמש נוסף בהצלחה

Enter לוחץ...

==== תפריט ניהול משתמשים ==
1. הוסף משתמש חדש
2. מחק משתמש
3. עדכן סיסמה
4. הצג רשימת משתמשים
5. התנתק

```

(1-5) בחר אפשרות : 4

:רשימת משתמשים

- admin
- gadi

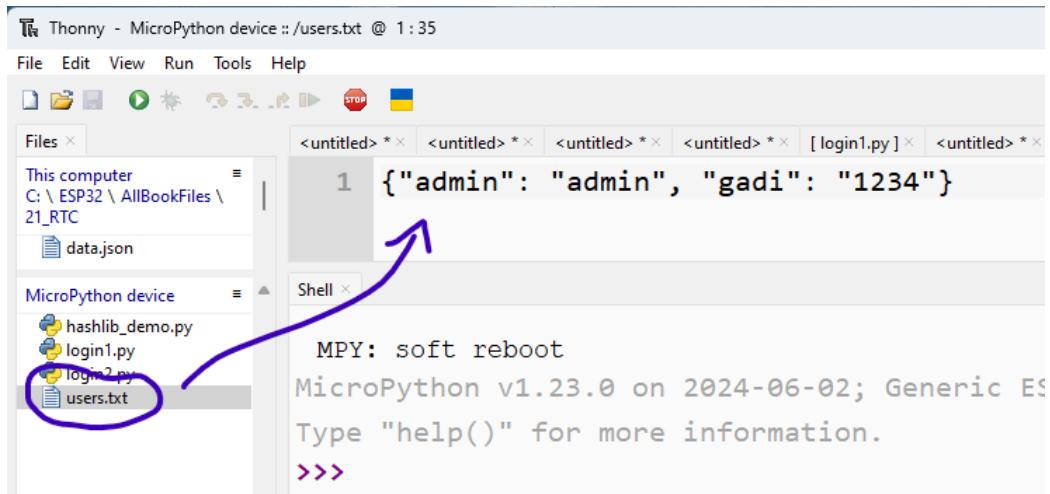
ללחוץ Enter לוחץ...

==== תפריט ניהול משתמשים ==

1. הוסף משתמש חדש
2. מחק משתמש
3. עדכן סיסמה
4. הצג רשימת משתמשים
5. התנתק

(1-5) בחר אפשרות : 5
!להתראות!

כמו כן ניתן לראות שנוצר לנו בברק קובץ חדש בשם users.txt המכיל את תוכן הבא:



כדי לפרק את בעיית הפריצה לפרויקט על ידי קריית תוכן הקובץ users.txt אנו נצפין את הסיסמה. נדגים תחילה את עקרון ה党校ה על ידי הקוד הבא:

```
# importing hashlib for getting sha256() hash function
import hashlib

def _bytes_to_hex(bytes_data):
    # A list containing all hexadecimal values
    hex_values = []
    # Go through each byte in the data
    for byte in bytes_data:
        # Converts each byte to a hexadecimal string with a fixed length of
2 characters
        hex_value = '{:02x}'.format(byte)
        # Adds the value to the list
        hex_values.append(hex_value)
    # concatenates all hexadecimal values into a single string
    s=''
    result = s.join(hex_values)
    return result

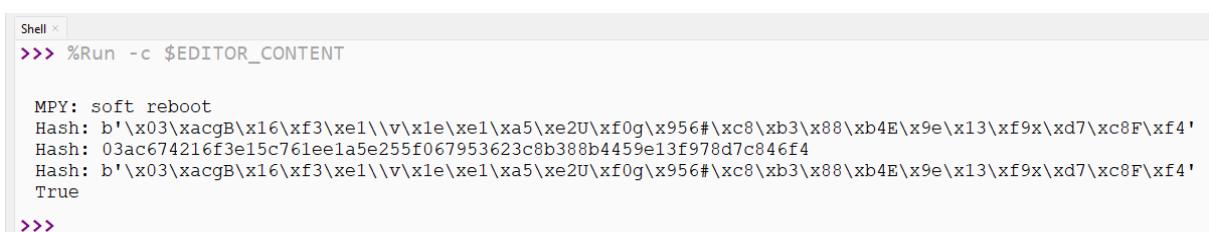
def _hex_to_bytes(hex_str):
    return bytes.fromhex(hex_str)

# A string that has been stored as a byte stream (due to the prefix b)
string = "1234"
# Initializing the sha256() method and passing the byte stream as an
argument
sha256 = hashlib.sha256(string)
# Hashes the data, and returns the output in hexadecimal format
string_hash = sha256.digest()
print("Hash:",string_hash)
string_hash1 = _bytes_to_hex(string_hash)
print("Hash:",string_hash1)
string_hash2 = _hex_to_bytes(string_hash1)
print("Hash:",string_hash2)

newstring = "1234"
newsha256 = hashlib.sha256(string)
newstring_hash = newsha256.digest()

print(newstring_hash == string_hash)
```

נקבל את הפלט הבא:



```
Shell < %
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Hash: b'\x03\xacgB\x16\xf3\xe1\\v\xle\xe1\x a5\xe2U\xf0g\x956#\xc8\xb3\x88\xb4E\x9e\x13\xf9x\xd7\xc8F\xf4'
Hash: 03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4
Hash: b'\x03\xacgB\x16\xf3\xe1\\v\xle\xe1\x a5\xe2U\xf0g\x956#\xc8\xb3\x88\xb4E\x9e\x13\xf9x\xd7\xc8F\xf4'
True
>>>
```

ניתן לראות שהסיסמה "1234" הפכה להיות הקוד הבא:

b'\x03\xacgB\x16\xf3\xe1\\v\x1e\xe1\xa5\xe2U\xf0g\x956#\xc8\xb3\x88\xb4E\x9e\x13\xf9x\xd7\xc8F\xf4'

כדי לשמר את הקוד זהה בקובץ אנו נעשה עליו המרה למחוזת הכלול ערכי טקסט בפורמט של מספרים הקסדצימליים באופן הבא:

03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4

ניתן לראות שהשורות זהות פרט לכך שהמירה הראשונה יצרה לנו מערך בתים הקסדצימלי והשורה שנייה יצרה לנו מחוזת טקסט זהה למערך שאויה ניתן לשמר בקובץ טקסט.

הצפנה sha256 היא הצפנה חד ציוונית כלומר הסיסמה "1234" תמיד תיצור את הקוד "03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4" לשחרר לאחר מכן מחוזת ההצפנה את הסיסמה המקורי. באופן זה גם אם יכנסו זרים לקובץ users.txt הם יראו את מחוזת ההצפנה אך לא יוכל לשחרר את הסיסמה המקורי כדי להתחבר בהצלחה.

תהליך ה- LOGIN מתבצע באופן המודגם בקוד זה:

```
newstring = "1234"
newsha256 = hashlib.sha256(string)
newstring_hash = newsha256.digest()

print(newstring_hash == string_hash)
```

כדי לבדוק האם הסיסמה שהמשתמש כתב זהה לזה השמורה בקובץ. אנו מבצעים פעולה הצפנה של הסיסמה שהמשתמש הקlid ומשווים בין ההצפנה של הקוד שהמשתמש כתב לזה השמורה בקובץ אם הקוד זהה, משמעות הדבר היא שהמשתמש כתב את הסיסמה המקורי נכון. אחרת הסיסמות לא נכונות!!

להלן שדרוג של הקוד up1.py כך שייכיל הפעם הצפנה של הסיסמה:

```
import json
import os
import hashlib

def _bytes_to_hex(bytes_data):
    hex_values = []
    for byte in bytes_data:
        hex_value = '{:02x}'.format(byte)
        hex_values.append(hex_value)
    result = ''.join(hex_values)
    return result

def encrypt_password(password):
    sha256 = hashlib.sha256(password)
    string_hash = sha256.digest()
    string_hash_hex = _bytes_to_hex(string_hash)
    return string_hash_hex

class UserManager:
    def __init__(self, filename="users.txt"):
        self.filename = filename
        self._ensure_file_exists()
```

```

def _ensure_file_exists(self):
    """ יוצר קובץ משתמשים אם לא קיים """
    if not self.filename in os.listdir():
        with open(self.filename, 'w') as f:
            json.dump({}, f)

def _read_users(self):
    """ קורא את רשימת המשתמשים מהקובץ """
    with open(self.filename, 'r') as f:
        return json.load(f)

def _save_users(self, users):
    """ שומר את רשימת המשתמשים לקובץ """
    with open(self.filename, 'w') as f:
        json.dump(users, f)

def verify_login(self, username, password):
    """ מאמת פרטיהתחברות """
    users = self._read_users()
    return username in users and users[username] == password

def add_user(self, username, password):
    """ מוסיף משתמש חדש """
    users = self._read_users()
    if username in users:
        return False, "שם המשתמש כבר קיים"
    users[username] = password
    self._save_users(users)
    return True, "המשתמש נוסף בהצלחה"

def delete_user(self, username):
    """ מוחק משתמש """
    users = self._read_users()
    if username not in users:
        return False, "המשתמש לא קיים"
    del users[username]
    self._save_users(users)
    return True, "המשתמש נמחק בהצלחה"

def update_password(self, username, new_password):
    """ מעדכן סיסמה למשתמש """
    users = self._read_users()
    if username not in users:
        return False, "המשתמש לא קיים"
    users[username] = encrypt_password(new_password)
    self._save_users(users)
    return True, "הסיסמה עודכניתה בהצלחה"

def list_users(self):
    """מחזיר רשימת משתמשים """
    return list(self._read_users().keys())

def admin_menu(user_manager):
    """תפריט ניהול משתמשים """

```

```

while True:
    print("\n====")
    print("1. ניהול משתמשים")
    print("2. מחיק משתמש")
    print("3. עדכן סיסמה")
    print("4. הציג רשימת משתמשים")
    print("5. הtgtנתק")

choice = input("\n1-5) : (בחר אפשרות ) ")

if choice == "1":
    username = input("הכנס שם משתמש חדש ")
    password = input("הכנס סיסמה ")
    success, message = user_manager.add_user(username,
encrypt_password(password))
    print(message)

elif choice == "2":
    username = input("הכנס שם משתמש למחיקה ")
    success, message = user_manager.delete_user(username)
    print(message)

elif choice == "3":
    username = input("הכנס שם משתמש ")
    new_password = input("הכנס סיסמה חדשה ")
    success, message = user_manager.update_password(username,
new_password)
    print(message)

elif choice == "4":
    users = user_manager.list_users()
    print("רשימת משתמשים:")
    for user in users:
        print(f"- {user}")

elif choice == "5":
    print("!להתראות")
    break

else:
    print("אפשרות לא חוקית")

input("... להמשך לחץ Enter")

def main():
    user_manager = UserManager()

    # אם אין משתמשים, צור משתמש ברירת מחדל
    if not user_manager.list_users():
        user_manager.add_user("admin", encrypt_password("admin"))
        print("נוצר משתמש ברירת מחדל: admin/admin")

    while True:
        print("\n====")

```

```

username = input("שם משתמש: ")
password = input("סיסמה: ")

if user_manager.verify_login(username, encrypt_password(password)):
    print("התחברת בהצלחה!")
    admin_menu(user_manager)
    break
else:
    print("שם משתמש או סיסמה שגויים")
    retry = input("לנסות שוב? (Y/N): ")
    if retry.lower() != 'y':
        break

if __name__ == "__main__":
    main()

```

חשוב! יש למחוק את הקובץ users.txt לפני הרצת קוד זה!

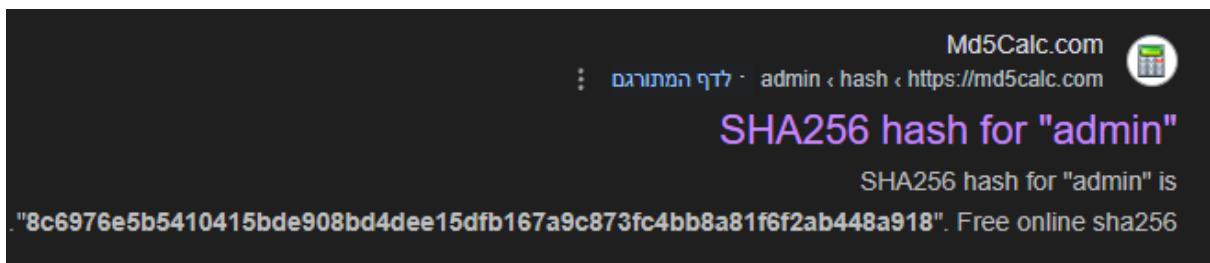
כasher נריץ את הקוד הבא נקבל את אותו התפריט כמו בדוגמה הקודמת אך הפעם הקובץ users.txt יראה כך:

```
{
"admin": "8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918",
"gadi": "03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4"
}
```

אתגר!

חיפוש קצר באינטרנט יגלה לכם שהמחרוזת `admin` היא `8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918`

לדוגמה:



כדי להתמודד גם עם אתגר זה ניתן ליצור קודים יותר מורכבים המבוססים על אותן עקרונות אך הפעם נosis לסיסמה מחרוזת טקסט אקראיית באופן הבא:

```

import json
import os
import hashlib

class PasswordEncryption:
    """מחלקה לטיפול בהצפנה סיסמאות"""

    @staticmethod

```

```

def _bytes_to_hex(bytes_data):
    """להציג הקסדיימליות מהיר"""
    return ''.join('{:02x}'.format(x) for x in bytes_data)

@staticmethod
def _hex_to_bytes(hex_str):
    """מחזיר מחרוזת הקסדיימלית ל-bytes"""
    return bytes.fromhex(hex_str)

@staticmethod
def encrypt_password(password):
    """מצפין סיסמה"""
    # אקרים salt יצירתי
    salt = os.urandom(4) # 4 bytes של salt
    # ונוספת ה-bytes-salt
    salted = password.encode() + salt
    # מהסיסמה hash יצירתי
    hashed = hashlib.sha256(salted).digest()
    # והמרה למחרוזת הקסדיימלית ה-hash-salt
    combined = salt + hashed
    return PasswordEncryption._bytes_to_hex(combined)

@staticmethod
def verify_password(password, encrypted):
    """אמת סיסמה מול גרסה מוצפנת"""
    try:
        # המירה לחזרה ל-bytes
        stored_data = PasswordEncryption._hex_to_bytes(encrypted)
        # בשים ראשוניים (4)-salt חילוץ ה
        salt = stored_data[:4]
        # המקורי hash
        stored_hash = stored_data[4:]
        # חדש מהסיסמה שהוכנעה hash יצירתי
        salted = password.encode() + salt
        new_hash = hashlib.sha256(salted).digest()
        # אם-hash השוואת ה
        return stored_hash == new_hash
    except:
        return False

class UserManager:
    def __init__(self, filename="users.txt"):
        self.filename = filename
        self._ensure_file_exists()

    def _ensure_file_exists(self):
        """יוצר קובץ משמשים אם לא קיים"""
        if not self.filename in os.listdir():
            with open(self.filename, 'w') as f:
                json.dump({}, f)

    def _read_users(self):
        """קורא את רשימת המשמשים מהקובץ"""
        with open(self.filename, 'r') as f:

```

```

        return json.load(f)

    def _save_users(self, users):
        """שומר את רשימת המשתמשים לקובץ"""
        with open(self.filename, 'w') as f:
            json.dump(users, f)

    def verify_login(self, username, password):
        """מאמת פרטיהתחברות"""
        users = self._read_users()
        if username not in users:
            return False
        return PasswordEncryption.verify_password(password,
users[username])

    def add_user(self, username, password):
        """מוסיף משתמש חדש"""
        users = self._read_users()
        if username in users:
            return False, "שם המשתמש כבר קיים"
        #הצפנה הסיסמה לפני השמירה
        encrypted_password = PasswordEncryption.encrypt_password(password)
        users[username] = encrypted_password
        self._save_users(users)
        return True, "המשתמש נוסף בהצלחה"

    def delete_user(self, username):
        """מחוקק משתמש"""
        users = self._read_users()
        if username not in users:
            return False, "המשתמש לא קיים"
        del users[username]
        self._save_users(users)
        return True, "המשתמש נמחק בהצלחה"

    def update_password(self, username, new_password):
        """מעדכן סיסמה למשתמש"""
        users = self._read_users()
        if username not in users:
            return False, "המשתמש לא קיים"
        #הצפנה הסיסמה החדשה לפני השמירה
        encrypted_password =
PasswordEncryption.encrypt_password(new_password)
        users[username] = encrypted_password
        self._save_users(users)
        return True, "הסיסמה עודכנה בהצלחה"

    def list_users(self):
        """מחזיר רשימת משתמשים"""
        return list(self._read_users().keys())

    def admin_menu(user_manager):
        """תפריט ניהול משתמשים"""
        while True:

```

```

print("\n==== תפריט ניהול משתמשים ===")
print("1. הוסף משתמש חדש")
print("2. מחק משתמש")
print("3. עדכן סיסמה")
print("4. הצג רשימת משתמשים")
print("5. הtgtntek")

choice = input("\n1-5) : (בחר אפשרות )"

if choice == "1":
    username = input("הכנס שם משתמש חדש ")
    password = input("הכנס סיסמה ")
    success, message = user_manager.add_user(username, password)
    print(message)

elif choice == "2":
    username = input("הכנס שם משתמש למחיקה ")
    success, message = user_manager.delete_user(username)
    print(message)

elif choice == "3":
    username = input("הכנס שם משתמש ")
    new_password = input("הכנס סיסמה חדשה ")
    success, message = user_manager.update_password(username,
new_password)
    print(message)

elif choice == "4":
    users = user_manager.list_users()
    print("\nרשימת משתמשים:")
    for user in users:
        print(f"- {user}")

elif choice == "5":
    print("!להתראות")
    break

else:
    print("אפשרות לא חוקית")

input("\nEnter לחץ...")

def main():
    user_manager = UserManager()

    # אם אין משתמשים, צור משתמש ברירת מחדל
    if not user_manager.list_users():
        user_manager.add_user("admin", "admin")
        print("נוצר משתמש ברירת מחדל admin/admin")

    while True:
        print("\n==== מערכת הת短时间内 ===")
        username = input("שם משתמש ")
        password = input("סיסמה ")

```

```
if user_manager.verify_login(username, password):
    print("ההתחברת בהצלחה!")
    admin_menu(user_manager)
    break
else:
    print("שם משתמש או סיסמה שגויים")
    retry = input("לנסות שוב? (Y/N) : ")
    if retry.lower() != 'y':
        break

if __name__ == "__main__":
    main()
```

משימה 28 - תאורה רצף נורוות ל- NeoPixel

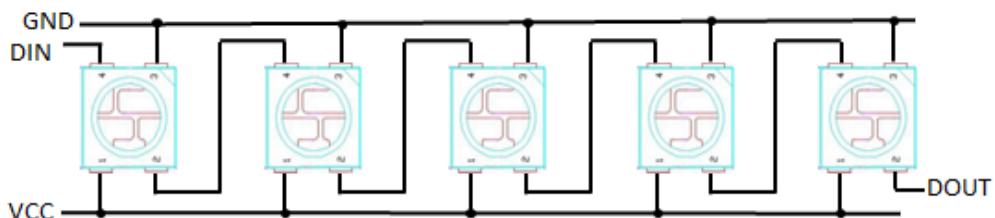
קישורים:

<https://docs.micropython.org/en/latest/esp8266/tutorial/neopixel.html#>

NeoPixel הוא פס LED, שבו כל נורה ניתנת לשיליטה באופן עצמאי, כך שניתן לשלוט בצבע ובהירות של כל נורה בנפרד. כל פס של NeoPixel מורכב ממספר נורות LED, שכל אחת מהן כוללת את רכיבי האור האדום, הירוק והכחול (RGB), ונitin לשילוט על כל אחד מהם באופן נפרד.

תכונות מרכזיות:

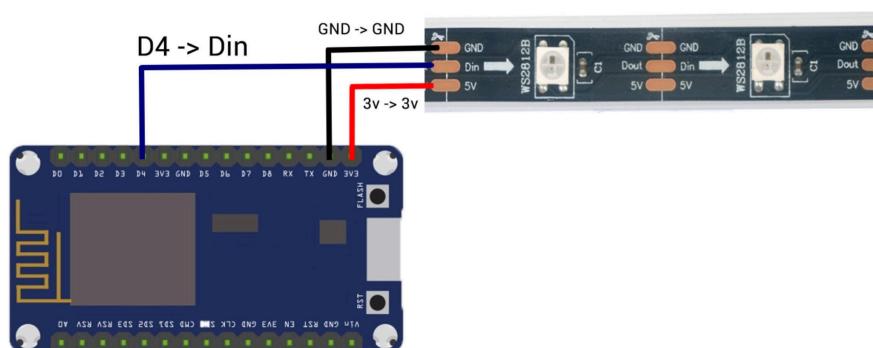
- כל נורה ב-NeoPixel ניתנת לשיליטה על ידי פלט נתונים שmagע מהבקר, מה שמאפשר ייצור אפקטים צבעוניים.
- הנתונים מעברים לנורות לפי סדר (אחד אחרי השני) כך שכל נורה מקבלת מידע, כל זאת תוך שימוש בהדק נתונים בודד בשם DIN.
- כל נורה יכולה להציג צבע אחד מתוך 16 מיליון צבעים אפשריים (RGB) ולכבות את האור כאשר יש צורך.
- ניתן לחבר כמה נורות NeoPixel אחת אחרי השנייה כך שהן יקבלו אותן רכיפים ויפעלו כרצף אחד.
- NeoPixel צריך חשמל בהתאם לצבעים שהוא מפיק, באופן כללי צבעים בהירים (למשל, צבעים לבנים) זקוקים יותר זרם מאשר צבעים כהים.



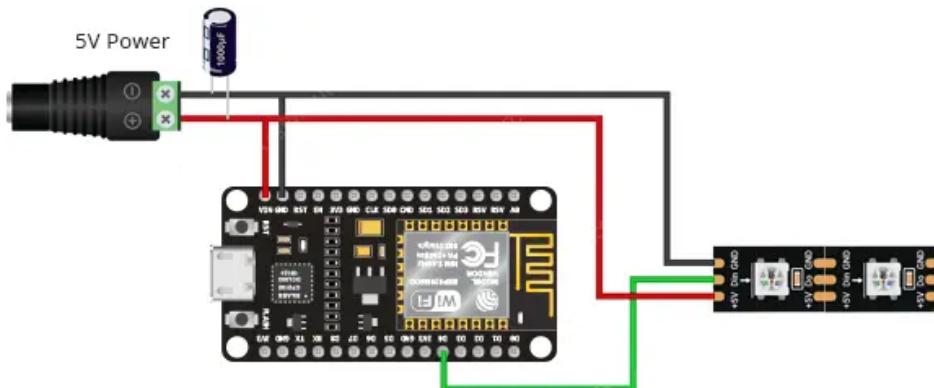
חיבור רכיב NeoPixel לבקר ESP32:

על מנת לחבר את רכיב NeoPixel לבקר ESP32, יש לחבר את ההדקים הבאים:

1. **VCC** - יש לחבר את קו ה-VCC של NeoPixel למתח של 5V (או 3.3V, תלוי בתנאים הטכניים של NeoPixel).
2. **GND** - יש לחבר את קו ה-GND של NeoPixel לפין GND של ESP32.
3. **DIN - Data Input** - יש לחבר את קו ה-DIN של NeoPixel לאינטראקציית>Digital I/O של בקר ESP32 (לרוב GPIO 5 או 4). זהו הקו דרכו עבירו נתונים השיליטה על הצבעים.



חיבור ללא מקור מתח חיצוני (לא מומלץ!)



חיבור כולל מקור מתח חיצוני (מומלץ!)

עקרון הפעולה

הברker NeoPixel מקבל את הנתונים בצורה טורית, סיבית אחר סיבית. כל נורה בשרשרת מקבלת את הנתונים שלא לפני סדר, כשהיא "יודעת" להפעיל את הצבע המבוקש על פי האות שהתקבל. הקידוד של הצבעים נעשה בצורה של סדרת פקודות, כאשר כל צבע מקודד בעזרת מספר ביטים באופן הבא:

- כל נורה מקבלת שלושה ערכים: אדום (R), ירוק (G), וכחול (B).
- כל ערך מצוין על ידי 8 ביטים (0-255).
- הנתונים מועברים בטור, וכך כל נורה מעביר את הנתונים לנורה הבאה.

הברker ESP32 שולח את הנתונים באמצעות אות ספרתי רציף, שמתחל בנורה הראשונה בשרשרת. לאחר מכן, כל נורה בשרשרת מקבלת את ערכי הצבע שלה, לבסוף נשלח סימן לנוריות להידלק כל אחת בהתאם לנiton שהיא קיבלה.

קוד דוגמה בשפת MicroPython לברker ESP32:

להלן דוגמת בסיסית להדלקת נוריות לד בודדה תוך שימוש בספרייה NeoPixel בספרייה mahahaha המובנת בשפת MicroPython. כך שאין צורך להתקין ספרייה כדי להפעיל את נוריות NeoPixel.

```
from machine import Pin
from neopixel import NeoPixel
from time import sleep

np = NeoPixel(Pin(14), 1)

np[0] = (0, 255, 255)
np.write()

sleep(1)

np[0] = (255, 0, 255)
np.write()

sleep(1)
```

```

np[0] = (255, 255, 0)
np.write()

sleep(1)

np[0] = (0, 0, 0)
np.write()

```

נדגים שימוש ב-8 נוריות neopixel הנראות כך:



דוגמה :1

```

from machine import Pin
from neopixel import NeoPixel
from time import sleep

np = NeoPixel(Pin(14), 8)

np[0] = (0, 255, 255)
np[1] = (255, 0, 255)
np[2] = (255, 255, 0)
np[3] = (0, 0, 255)
np[4] = (255, 0, 0)
np[5] = (0, 255, 0)
np[6] = (128, 128, 20)
np[7] = (128, 0, 200)
np.write()

sleep(2)

for i in range(8):
    np[i] = (0, 0, 0)
np.write()

```

דוגמה :2

```

from machine import Pin
from neopixel import NeoPixel
from time import sleep_ms

```

```

np = NeoPixel(Pin(14), 8)
n = np.n

for i in range(100):
    for j in range(n):
        np[j] = (0, 0, 0)
    np[i % n] = (255, 255, 255)
    np.write()
    sleep_ms(25)

for i in range(n):
    np[i] = (0, 0, 0)
np.write()

```

:3 דוגמה

```

from machine import Pin
from neopixel import NeoPixel
from time import sleep_ms

np = NeoPixel(Pin(14), 8)
n = np.n

for i in range(100):
    for j in range(n):
        np[j] = (0, 0, 128)
    if (i // n) % 2 == 0:
        np[i % n] = (0, 0, 0)
    else:
        np[n - 1 - (i % n)] = (0, 0, 0)
    np.write()
    sleep_ms(60)

for i in range(n):
    np[i] = (0, 0, 0)
np.write()

```

:4 דוגמה

```

from machine import Pin
from neopixel import NeoPixel
from time import sleep_ms

np = NeoPixel(Pin(14), 8)
n = np.n

for i in range(0, 4 * 256, 8):
    for j in range(n):
        if (i // 256) % 2 == 0:
            val = i & 0xff
        else:
            val = 255 - (i & 0xff)
        np[j] = (val, 0, 0)
    np.write()
    sleep_ms(10)

```

```

        np[j] = (val, 0, 0)
        np.write()

for i in range(n):
    np[i] = (0, 0, 0)
np.write()

```

דוגמה 5

```

from machine import Pin, reset
from neopixel import NeoPixel
from time import sleep

rainbow = [
    (126, 1, 0), (114, 13, 0), (102, 25, 0), (90, 37, 0), (78, 49, 0),
    (66, 61, 0), (54, 73, 0), (42, 85, 0),
    (30, 97, 0), (18, 109, 0), (6, 121, 0), (0, 122, 5), (0, 110, 17),
    (0, 98, 29), (0, 86, 41), (0, 74, 53),
    (0, 62, 65), (0, 50, 77), (0, 38, 89), (0, 26, 101), (0, 14, 113),
    (0, 2, 125), (9, 0, 118), (21, 0, 106),
    (33, 0, 94), (45, 0, 82), (57, 0, 70), (69, 0, 58), (81, 0, 46), (93,
    0, 34), (105, 0, 22), (117, 0, 10)]
]

try:
    print("Press Ctrl-C to Cleaning up and exiting...")
    np = NeoPixel(Pin(14), 8)
    n = np.n
    while True:
        rainbow = rainbow[-1:] + rainbow[:-1]
        for i in range(n):
            np[i] = rainbow[i]
        np.write()
        sleep(0.5)
except KeyboardInterrupt:
    print("\nCtrl-C pressed. Cleaning up and exiting...")
finally:
    for i in range(n):
        np[i] = (0, 0, 0)
    np.write()
    reset()

```

דוגמה 6

* תודה ליאוב גולן על הרעיון לכתיבה הפרק ועל הקוד מצ"ב

```

import machine, neopixel
from time import sleep
from random import randint

np = neopixel.NeoPixel(machine.Pin(14), 8)

while True:
    for i in range(8):

```

```

np[i] = (randint(0,255), randint(0,255), randint(0,255))
np.write()
sleep(0.01)
for i in range (8):
    np[i] = (0, 0, 0)
    np.write()
for i in range (7,-1,-1):
    np[i] = (randint(0,255), randint(0,255), randint(0,255))
    np.write()
    sleep(0.01)
for i in range (8):
    np[i] = (0, 0, 0)
    np.write()

```

פרוטוקול תקשורת WS2812W - מדריך טכני

פרוטוקול WS2812W הוא פרוטוקול תקשורת טורי חד-כיווני המשמש לשיליטה בנוורות LED RGB מסוג NeoPixel.

הפרוטוקול מאפשר שליטה בשרשראת של נורות LED באמצעות קוו נתונים ייחיד.

מאפיינים טכניים:

- סדר צבעים: GRB (ירוק, אדום, כחול)
- 24 ביט סה"כ
- כל צבע 8 סיביות, משודר מסיבת MSB לכיוון סיבית LSB, כלומר הביט המשמעותי ביותר (MSB) נשלח ראשון
 - כל LED מקבל 24 ביט של מידע (3 בתים)
 - סדר השליחה: ירוק (8 ביט) -> אדום (8 ביט) -> כחול (8 ביט)
 - זמן איפוא <50μs

Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Note: Follow the order of GRB to sent data and the high bit sent at first.

שידור ביט '0' לוגי:

- אוט גבוה: $0.35\mu s \pm 150ns$
- אוט נמוך: $0.8\mu s \pm 150ns$

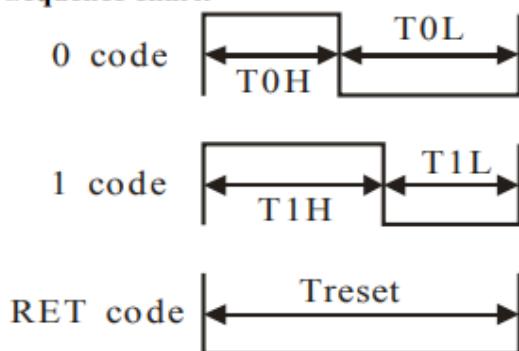
שידור ביט '1' לוגי:

- אוט גבוה: $0.7\mu s \pm 150ns$
- אוט נמוך: $0.6\mu s \pm 150ns$

Data transfer time(TH+TL=1.25μs±600ns)

T0H	0 code ,high voltage time	0.35us	$\pm 150\text{ns}$
T1H	1 code ,high voltage time	0.7us	$\pm 150\text{ns}$
T0L	0 code , low voltage time	0.8us	$\pm 150\text{ns}$
T1L	1 code ,low voltage time	0.6us	$\pm 150\text{ns}$
RES	low voltage time	Above 50μs	

Sequence chart:



שרשרת הנתונים

- כל LED מעביר את הנתונים הנוגעים ל-LED הבא
- ה-LED הראשון לוקח את 24 הביטים הראשונים
- שאר הנתונים ממשיכים בשרשראת

אות איפוס (Reset)

- נדרש אות נמוך של לפחות 50μs בין חבילות נתונים
- מופיע את כל הנורות בשרשראת ומכוון אותו לקבלת נתונים חדשים

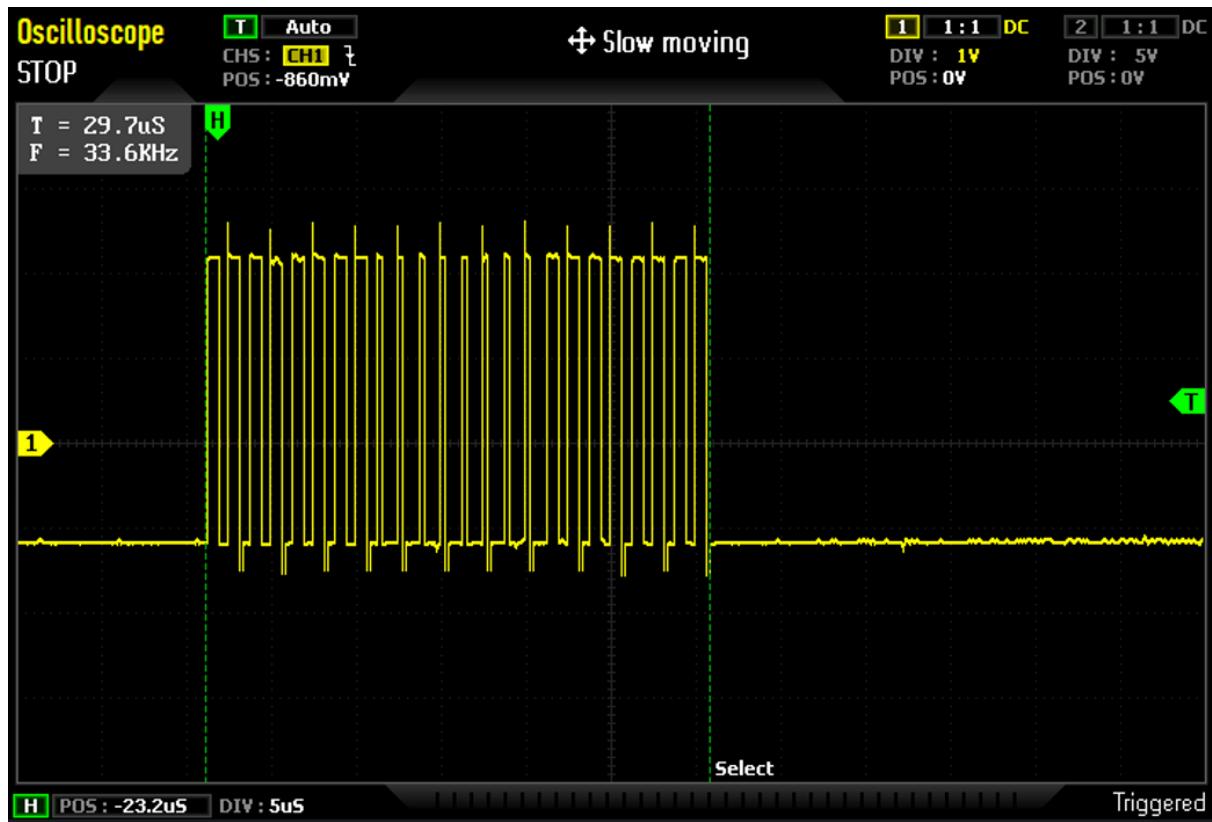
נבחן את המאפיינים הנ"ל מניתוח שידור אות על סקופ בהתאם לקוד הבא:

```
from machine import Pin
from neopixel import NeoPixel

np = NeoPixel(Pin(14), 1)
while True :
    np[0] = (0, 255, 255)
    np.write()
```

שימוש לב שלא מדובר כאן בקוד שמייצר פלט תאורה מעניין, מדובר בקוד שמייצר באופן מחזורי שידור של אות ברור יחסית כך שניתן יהיה לדגום אותו בעזרת סקופ.

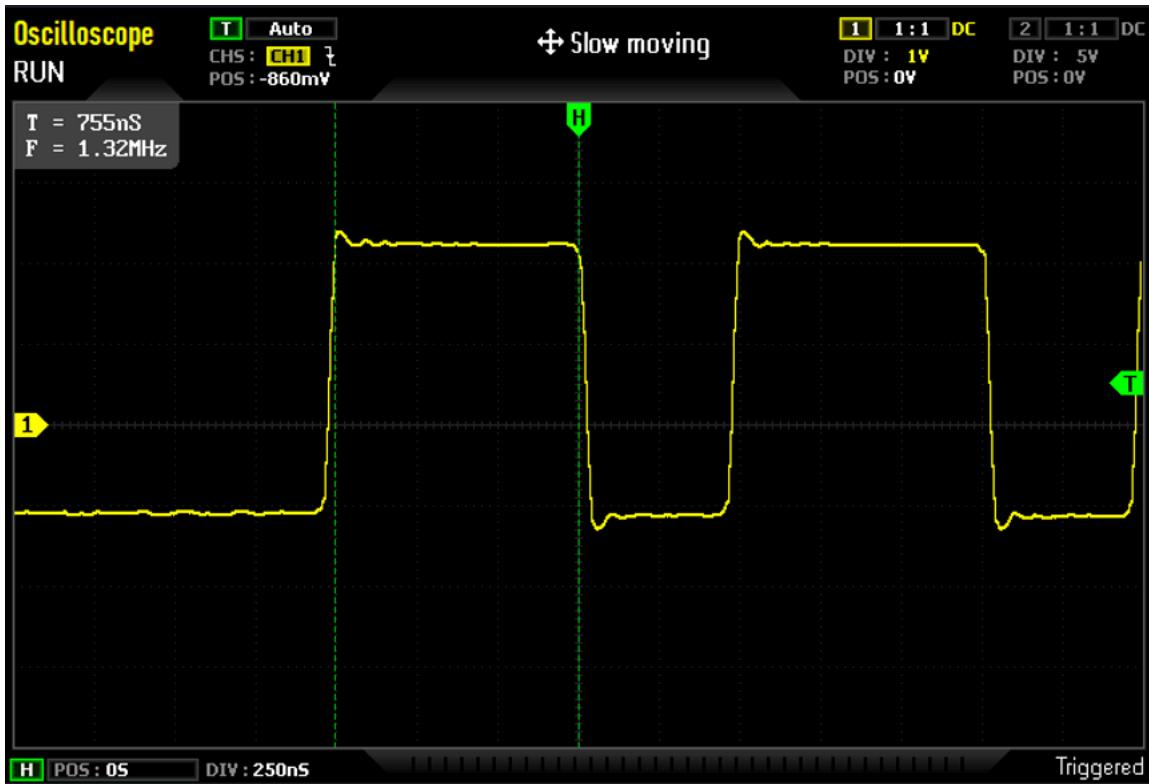
להלן הפלט המתkeletal:



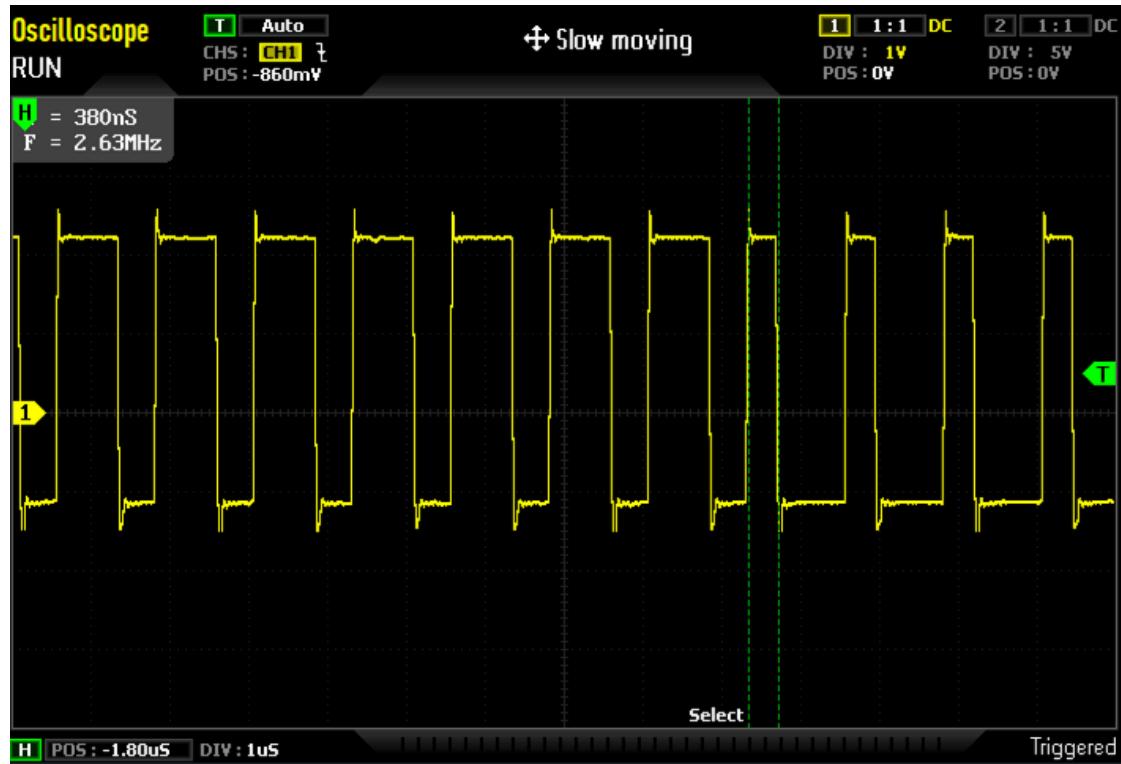
ניתן לראות שידור של רצף 24 סיביות המיעדים להדלקת פיקסל בודד, כאשר 8 אחדים במשר 8 אפסים ואז 8 אחדים כמתואר בקוד הבא:

```
np[0] = (0, 255, 255)
np.write()
```

נבחן את זמני האות עצמה:



ניתן לראות דוגמה לשידור אחד לוגי כך שזמן האות בرمה גבוהה הוא 755ns כאשר היצרן מצין שהוא צריך להיות $\text{ns} 150 \pm 0.7$ מה שעומד בסטנדרט.



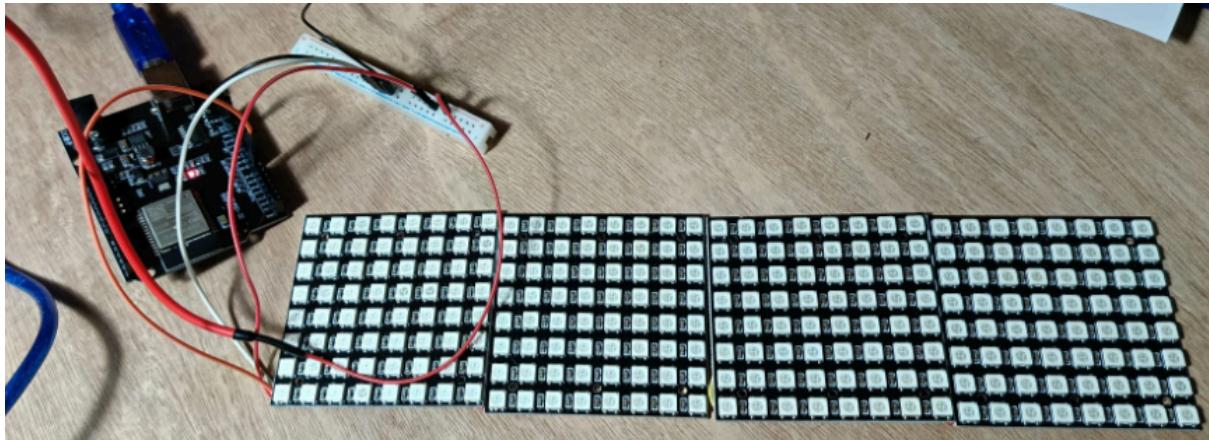
ניתן לראות דוגמה לשידור אפס לוגי כך שזמן האות בرمה גבוהה הוא 380ns כאשר היצרן מצין שהוא צריך להיות $\text{ns} 150 \pm 0.35$ מה שעומד בסטנדרט.

משימה 29 - הציג טקסט על גבי מטריצות 8x8 עם MicroPython ב-ESP32

קישורים:

https://github.com/GadiHerman/ESP32_MicroPython_AllBookFiles/blob/main/28_NeoPixel_TextProject/TextOnNeoPixel3.py

במשימה זו נלמד כיצד להציג טקסט על גבי 4 מטריצות של NeoPixel (כל אחת בגודל 8x8) המוחוברות יחד ליצירת תצוגה גדולה יותר, תוך שימוש בקוד MicroPython עבור בקר ESP32.



ציוד נדרש:

- בקר ESP32
- 4 מטריצות 8x8 NeoPixel (סה"כ 256 LED)
- מוקור מתאים (5V עם זרם מספיק)
- כבלי חיבור
- מחשב עם תוכנת Thonny או IDE אחרת לתוכנות MicroPython

הכנות החומרה:

- מחברים את המטריצות בשרשראת (יציאת DATA של מטריצה אחת מחוברת לכינית DATA של הבאה אחרת)
- מחברים את המטריצה הראשונה לפין 14 ב-ESP32 (ניתן לשנות לפי הצורך)
- מספקים מתח 5V יציב לכל המטריצות מספק חיצוני !!! שימוש לב הזרם שהמטריצה צריכה גדול לאין שיעור ממה שיוכלו לספק הבקר עצמו.



- מחברים את כל האדמהות (GND) יחד.

הבנת הקוד:

הקוד מספק מחלקה בשם Matrix המטפלת בכל הfonקציוניות של התצוגה:

פרמטרים במחלקה:

- pin: מספר הידק אליו מחובר ה-NeoPixel הראשון
- width: רוחב כולל של כל המטריצות (32 פיקסלים ל-4 מטריצות)
- height: גובה המטריצות (8 פיקסלים)
- color: צבע ברירת מחדל [R, G, B]
- CharSpacing: ריבוע בין תווים (ברירת מחדל 1)

הפעולות העיקריות של המחלקה:

- clear: מכבה את כל הפיקסלים
- set_pixel: מדליק פיקסל ספציפי בצבע נתון
- show_char: מציג תו במקומות אופקיים מסוימים
- show_text: מציג טקסט עם אפשרות לגיליה

הגדרת התצוגה:

```
pin = 14
width = 32 # 4 32 = מטריצות של 8 פיקסלים
height = 8 # גובה כל מטריצה #
color1 = [255, 0, 0] # אדום
mat = Matrix(Pin(pin), width, height, color1)
```

הציג טקסט סטטי:

```
mat.show_text("12:45", scroll=False)
sleep(2)
```

הציג טקסט עם גלילה:

```
mat.show_text("ABC abc", scroll=True, scroll_delay=0.01, color=[0, 0, 255])
```

```
sleep(2)
```

שינוי צבעים:

ניתן להגדיר צבעים שונים עבור כל הודעה:

```
color2 = [0, 255, 0] # ירוק
mat.show_text("HELLO!", scroll=False, color=color2)
```

ניקי התצוגה:

```
mat.clear()
```

תרגול מעשי:

1. הפעילו את הקוד והציגו את שמכם על המטריצות

2. שנו את מהירות הגלילה על ידי שינוי פרמטר scroll_delay

3. נסו צבעים שונים על ידי שינוי ערכי ה-RGB

4. הוסיפו הודעות נוספות עם הגדרות שונות

טיפים חשובים:

1. וודאו שהספק המתח מספק מספיק זרם לכל המטריצות.

2. התחילו עם בהירות נמוכה (ערכי צבע קטנים מ-50) כדי למנוע עומס יתר

3. במידה ויש בעיות, בדקו את חיבור ה-**DATA** לבר

הרחבות אפשריות:

1. הוספת אнимציות מעבר בין הודעות

2. תמייה בתווים מיוחדים או סמלים מותאמים אישית

3. שליטה מרוחק בהודעות דרך WiFi

4. שילוב עם חיישנים להציג נתונים בזמן אמת

להלן הקוד המלא:

```
"""
ESP32 Micropython NeoPixel 8x32 matrix text display
```

https://github.com/GadiHerman/ESP32_MicroPython_AllBookFiles

MIT License

Copyright (c) 2025 Gadi Herman

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

"""

```
from machine import Pin
from neopixel import NeoPixel
from time import sleep

class Matrix:

    # Font definition - each character is defined as 5x7 bitmap
    FONT = [
        [0x00, 0x00, 0x00, 0x00, 0x00], # 0x20 space
        [0x00, 0x00, 0x5F, 0x00, 0x00], # 0x21 !
        [0x00, 0x07, 0x00, 0x07, 0x00], # 0x22 "
        [0x14, 0x7F, 0x14, 0x7F, 0x14], # 0x23 #
        [0x24, 0x2A, 0x7F, 0x2A, 0x12], # 0x24 $
        [0x23, 0x13, 0x08, 0x64, 0x62], # 0x25 %
        [0x36, 0x49, 0x56, 0x20, 0x50], # 0x26 &
        [0x00, 0x08, 0x07, 0x03, 0x00], # 0x27 '
        [0x00, 0x1C, 0x22, 0x41, 0x00], # 0x28 (
        [0x00, 0x41, 0x22, 0x1C, 0x00], # 0x29 )
        [0x2A, 0x1C, 0x7F, 0x1C, 0x2A], # 0x2A *
        [0x08, 0x08, 0x3E, 0x08, 0x08], # 0x2B +
        [0x00, 0x80, 0x70, 0x30, 0x00], # 0x2C ,
        [0x08, 0x08, 0x08, 0x08], # 0x2D -
        [0x00, 0x00, 0x60, 0x60, 0x00], # 0x2E .
        [0x20, 0x10, 0x08, 0x04, 0x02], # 0x2F /
        [0x3E, 0x51, 0x49, 0x45, 0x3E], # 0x30 0
        [0x00, 0x42, 0x7F, 0x40, 0x00], # 0x31 1
        [0x72, 0x49, 0x49, 0x49, 0x46], # 0x32 2
        [0x21, 0x41, 0x49, 0x4D, 0x33], # 0x33 3
        [0x18, 0x14, 0x12, 0x7F, 0x10], # 0x34 4
        [0x27, 0x45, 0x45, 0x45, 0x39], # 0x35 5
        [0x3C, 0x4A, 0x49, 0x49, 0x31], # 0x36 6
        [0x41, 0x21, 0x11, 0x09, 0x07], # 0x37 7
        [0x36, 0x49, 0x49, 0x49, 0x36], # 0x38 8
```

```

[0x46, 0x49, 0x49, 0x29, 0x1E], # 0x39 9
[0x00, 0x00, 0x14, 0x00, 0x00], # 0x3A :
[0x00, 0x40, 0x34, 0x00, 0x00], # 0x3B ;
[0x00, 0x08, 0x14, 0x22, 0x41], # 0x3C <
[0x14, 0x14, 0x14, 0x14, 0x14], # 0x3D =
[0x00, 0x41, 0x22, 0x14, 0x08], # 0x3E >
[0x02, 0x01, 0x59, 0x09, 0x06], # 0x3F ?
[0x3E, 0x41, 0x5D, 0x59, 0x4E], # 0x40 @
[0x7C, 0x12, 0x11, 0x12, 0x7C], # 0x41 A
[0x7F, 0x49, 0x49, 0x49, 0x36], # 0x42 B
[0x3E, 0x41, 0x41, 0x41, 0x22], # 0x43 C
[0x7F, 0x41, 0x41, 0x41, 0x3E], # 0x44 D
[0x7F, 0x49, 0x49, 0x49, 0x41], # 0x45 E
[0x7F, 0x09, 0x09, 0x09, 0x01], # 0x46 F
[0x3E, 0x41, 0x41, 0x51, 0x73], # 0x47 G
[0x7F, 0x08, 0x08, 0x08, 0x7F], # 0x48 H
[0x00, 0x41, 0x7F, 0x41, 0x00], # 0x49 I
[0x20, 0x40, 0x41, 0x3F, 0x01], # 0x4A J
[0x7F, 0x08, 0x14, 0x22, 0x41], # 0x4B K
[0x7F, 0x40, 0x40, 0x40, 0x40], # 0x4C L
[0x7F, 0x02, 0x1C, 0x02, 0x7F], # 0x4D M
[0x7F, 0x04, 0x08, 0x10, 0x7F], # 0x4E N
[0x3E, 0x41, 0x41, 0x41, 0x3E], # 0x4F O
[0x7F, 0x09, 0x09, 0x09, 0x06], # 0x50 P
[0x3E, 0x41, 0x51, 0x21, 0x5E], # 0x51 Q
[0x7F, 0x09, 0x19, 0x29, 0x46], # 0x52 R
[0x26, 0x49, 0x49, 0x49, 0x32], # 0x53 S
[0x03, 0x01, 0x7F, 0x01, 0x03], # 0x54 T
[0x3F, 0x40, 0x40, 0x40, 0x3F], # 0x55 U
[0x1F, 0x20, 0x40, 0x20, 0x1F], # 0x56 V
[0x3F, 0x40, 0x38, 0x40, 0x3F], # 0x57 W
[0x63, 0x14, 0x08, 0x14, 0x63], # 0x58 X
[0x03, 0x04, 0x78, 0x04, 0x03], # 0x59 Y
[0x61, 0x59, 0x49, 0x4D, 0x43], # 0x5A Z
[0x00, 0x7F, 0x41, 0x41, 0x41], # 0x5B [
[0x02, 0x04, 0x08, 0x10, 0x20], # 0x5C \
[0x00, 0x41, 0x41, 0x41, 0x7F], # 0x5D ]
[0x04, 0x02, 0x01, 0x02, 0x04], # 0x5E ^
[0x40, 0x40, 0x40, 0x40, 0x40], # 0x5F -
[0x00, 0x03, 0x07, 0x08, 0x00], # 0x60 .
[0x20, 0x54, 0x54, 0x78, 0x40], # 0x61 a
[0x7F, 0x28, 0x44, 0x44, 0x38], # 0x62 b
[0x38, 0x44, 0x44, 0x44, 0x00], # 0x63 c
[0x38, 0x44, 0x44, 0x28, 0x7F], # 0x64 d
[0x38, 0x54, 0x54, 0x54, 0x18], # 0x65 e
[0x00, 0x08, 0x7E, 0x09, 0x02], # 0x66 f
[0x0C, 0x52, 0x52, 0x4E, 0x3C], # 0x67 g
[0x7F, 0x08, 0x04, 0x04, 0x78], # 0x68 h
[0x00, 0x44, 0x7D, 0x40, 0x00], # 0x69 i
[0x20, 0x40, 0x40, 0x3D, 0x00], # 0x6A j
[0x7F, 0x10, 0x28, 0x44, 0x00], # 0x6B k
[0x00, 0x41, 0x7F, 0x40, 0x00], # 0x6C l
[0x7C, 0x04, 0x78, 0x04, 0x78], # 0x6D m
[0x7C, 0x08, 0x04, 0x04, 0x78], # 0x6E n
[0x38, 0x44, 0x44, 0x44, 0x38], # 0x6F o
[0xFC, 0x18, 0x24, 0x24, 0x18], # 0x70 p
[0x0C, 0x12, 0x12, 0x0C, 0x7E], # 0x71 q
[0x7C, 0x08, 0x04, 0x04, 0x08], # 0x72 r

```

```

[0x48, 0x54, 0x54, 0x54, 0x24], # 0x73 s
[0x04, 0x04, 0x3F, 0x44, 0x24], # 0x74 t
[0x3C, 0x40, 0x40, 0x20, 0x7C], # 0x75 u
[0x1C, 0x20, 0x40, 0x20, 0x1C], # 0x76 v
[0x3C, 0x40, 0x30, 0x40, 0x3C], # 0x77 w
[0x44, 0x28, 0x10, 0x28, 0x44], # 0x78 x
[0x0C, 0x10, 0x10, 0x50, 0x3C], # 0x79 y
[0x44, 0x64, 0x54, 0x4C, 0x44], # 0x7A z
[0x00, 0x08, 0x36, 0x41, 0x00], # 0x7B {
[0x00, 0x00, 0x7F, 0x00, 0x00], # 0x7C |
[0x00, 0x41, 0x36, 0x08, 0x00], # 0x7D }
[0x02, 0x01, 0x02, 0x04, 0x02] # 0x7E ~
]

def __init__(self, pin, width, height, color, CharSpacing=1):
    """
    Initialize Matrix object

    Args:
        pin: The GPIO pin to use
        width: Width of the matrix
        height: Height of the matrix
        color: Default color as [r, g, b]
    """
    self.width = width
    self.height = height
    self.color = color
    self.CharSpacing = CharSpacing
    self.First_Font_index = 0x20
    self.np = NeoPixel(pin, width * height)
    self.clear()

def clear(self):
    """Clear all pixels (set to black)"""
    for i in range(self.width * self.height):
        self.np[i] = (0, 0, 0)
    self.np.write()

def set_pixel(self, x, y, color=None):
    """
    Set a pixel at (x, y) to specified color

    Args:
        x: X coordinate (0 to width-1)
        y: Y coordinate (0 to height-1)
        color: Color as [r, g, b], uses default color if None
    """
    if x < 0 or x >= self.width or y < 0 or y >= self.height:
        return # Out of bounds

    if color is None:
        color = self.color

    index = x * self.height + y
    self.np[index] = tuple(color)

def show_char(self, char_data, x_offset=0, color=None):
    """

```

```

    Display a character at the specified offset

Args:
    char_data: Character data as 5 bytes (each representing a column)
    x_offset: X position to start from
    color: Color to use, uses default if None
"""
if color is None:
    color = self.color
char_map = self.FONT[ord(char_data)-self.First_Font_index]
# Draw the character (5 columns wide)
for col in range(5):
    byte = char_map[col]
    for row in range(7): # Font height is 7
        if byte & (1 << row): # Check if bit is set
            # 6-row to flip vertically
            self.set_pixel(x_offset + col, 6 - row, color)
self.np.write()

def show_text(self, text_data, scroll=True, scroll_delay=0.1, color=None):
"""
Display text on the matrix

Args:
    text_data: List of character data (each being 5 bytes)
    scroll: Whether to scroll the text
    scroll_delay: Delay between scroll steps
    color: Color to use, uses default if None
"""
if color is None:
    color = self.color

if not scroll and len(text_data) * (5+ self.CharSpacing) > self.width:
    # Can't fit on screen, force scrolling
    scroll = True

# Clear the display
self.clear()

if not scroll:
    # Static display
    x_offset = 0
    for char_data in text_data:
        self.show_char(char_data, x_offset, color)
        x_offset += 5 + self.CharSpacing # Char width + space
    self.np.write()
else:
    # Scrolling display
    # Start with text off the right edge
    total_width = len(text_data) * (5+ self.CharSpacing)

    for offset in range(self.width + total_width):
        self.clear()
        char_offset = 0

        for char_data in text_data:
            char_pos = self.width - offset + char_offset
            # Only draw if on screen

```

```

        if char_pos < self.width and char_pos > -4:
            self.show_char(char_data, char_pos, color)
        char_offset += 5 + self.CharSpacing # Char width + space

    self.np.write()
    sleep(scroll_delay)

# Example usage:
pin = 14
width = 32
height = 8
color1 = [255, 0, 0]
color2 = [0, 255, 0]
color3 = [0, 0, 255]
mat = Matrix(Pin(pin), width, height, color1)
mat.show_text("12:45", scroll=False)
sleep(2)
mat.show_text("HELLO!", scroll=False, color=color2)
sleep(2)
mat.show_text("Hello Gadi.", scroll=True, scroll_delay=0.01, color=color3)
sleep(2)
mat.clear()

```

משימה 30 - עבודה עם ערוץ תקשורת I2C ועבודה עם היישנים

קישורים:

https://github.com/GadiHerman/ESP32_MicroPython_AllBookFiles/blob/main/28_NeoPixel_TextProject/TextOnNeoPixel3.py

בפועלות זו נלמד כיצד להשתמש בעורך תקשורת C2I כדי לקרוא נתונים משני חישנים פופולריים: BMP280 (חישן לחץ וטמפרטורה) ו-AHT20 (חישן טמפרטורה ולחות). הפעולות כוללות הכרת פרוטוקול C2I, סריקת רכיבים וחיבור לספריות ספציפיות.

אידנדן

- מחשוב עם תוכנת פיתוח MicroPython
 - מקור מתח מתאים
 - כבלי חיבור (ג'אמפרים)
 - חיישן AHT20 או AHT10
 - חיישן BMP280
 - בקר ESP32

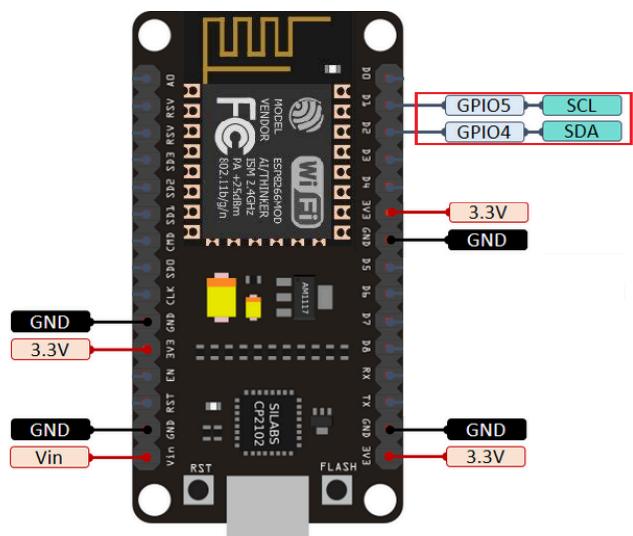
מבוא ל-12c

- פרוטוקול C2O הוא תקן תקשורת טורי למעגלים משולבים הכלול שימוש בשני חיוטים בלבד: ADA (נתונים) ו-SCL (שעון)
 - כל רכיב כולל כתובות ייחודית (7 סיביות בדרכן כלל)
 - תומך ב מהירותים שונות (100kHz, 400kHz, 1MHz)

יישום תקשורת I2C ב-ESP32

ניתן למש C20 בשתי דרכים:

- חומרה (I2C) - משתמש בمعالgi C2I מובנים של ה-ESP32
 - תוכנה (SoftI2C) - ממומש בתוכנה ופועל על הדקים גנריים



סרייקת רכיבים חומרה המתחברים על ערוץ I2C לבקר ESP32

סרייקת רכיבים תוך שימוש בחומרה בבקר:

```
from machine import I2C, Pin

i2c = I2C(scl=Pin(9), sda=Pin(8))

devices = i2c.scan()
if len(devices) == 0:
    print("No i2c device !")
else:
    print('i2c devices found:',len(devices))
for device in devices:
    print("At address: ",hex(device))
```

סרייקת רכיבים עם SoftI2C

```
from machine import SoftI2C, Pin

i2c = SoftI2C(scl=Pin(9), sda=Pin(8))

devices = i2c.scan()
if len(devices) == 0:
    print("No i2c device !")
else:
    print('i2c devices found:',len(devices))
for device in devices:
    print("At address: ",hex(device))
```

נקבל את הפלט הבא:

```
Shell >

MPY: soft reboot
Warning: I2C(-1, ...) is deprecated, use SoftI2C(...) instead
i2c devices found: 2
At address: 0x38
At address: 0x77

>>>
```

הבדלים עיקריים:

- I2C בחומרה - מהיר יותר אך מוגבל לפינים ספציפיים.
- SoftI2C - גמיש יותר, פועל על כל פין אך איטי יותר.

עבודה עם חיישן AHT20

- יש לוודא שהספרייה `adafruit_ahtx0` שמורה כקובץ בזיכרון של הרכיב

- החישן פועל בכתובת 0x38 (בדרך כלל)

The screenshot shows a file browser on the left and a code editor on the right. The file browser lists files in two locations: 'This computer' and 'MicroPython device'. In 'This computer', there is a folder '30_AHT20andBMP280' containing several Python files. In 'MicroPython device', there is a folder 'ahtx0' containing the same files. A blue arrow points from the 'ahtx0' folder in the file browser to the 'ahtx0.py' file in the code editor. The code editor displays the following Python script:

```
import utime
from machine import Pin, I2C
import ahtx0

i2c = I2C(scl=Pin(9), sda=Pin(8))

# Create the sensor object using I2C
sensor = ahtx0.AHT10(i2c)

while True:
    print("\nTemperature: %0.2f C" % sensor.temperature)
    print("Humidity: %0.2f %%" % sensor.relative_humidity)
    utime.sleep(5)
```

להלן קוד קריית נתונים טמפרטורה ולחות מהרכיב:

```
import utime
from machine import Pin, I2C
import ahtx0

i2c = I2C(scl=Pin(9), sda=Pin(8))

# Create the sensor object using I2C
sensor = ahtx0.AHT10(i2c)

while True:
    print("\nTemperature: %0.2f C" % sensor.temperature)
    print("Humidity: %0.2f %%" % sensor.relative_humidity)
    utime.sleep(5)
```

הסביר:

1. יוצרים עצם I2C עם הפינים המתאימים

2. יוצרים עצם חישן מהמספריה 0 ahtx0

3. קוראים את הנתונים במרווח זמן קבועים

עבודה עם חיישן BMP280

- נדרשת ספריתת py
- החישן פועל בכתובת 0x76 או 0x77 (תלוי בחיווט)

להלן קוד לקריאת לחץ וטמפרטורה מהחישן

```
from BMP280 import *
#from machine import SoftI2C, Pin
from machine import I2C, Pin

#i2c = SoftI2C(scl=Pin(9), sda=Pin(8))
i2c = I2C(0, scl = Pin(9), sda = Pin(8), freq = 400000)

bmp = BMP280(i2c, addr=0x77)

bmp.use_case(BMP280_CASE_INDOOR)
bmp.oversample(BMP280_OS_ULTRAHIGH)

bmp.temp_os = BMP280_TEMP_OS_8
bmp.press_os = BMP280_PRES_OS_8

bmp.standby = BMP280_STANDBY_250
bmp.iir = BMP280_IIR_FILTER_2

bmp.force_measure()

print(bmp.temperature)
print(bmp.pressure/100.0)
```

הסבר:

1. הגדרת ערוץ תקשורת I2C ב מהירות 400kHz
2. יצירת עצם חיישן עם הכתובת המתאימה
3. הגדרת פרמטרים אופטימליים למדידה, על פי נתוני הייצור.
4. הפעלת מדידה וקריאת התוצאות

תרגול מעשי:

1. חיבור פיזי:

- חיבור את שני החישנים ל-ESP32 באותו ערוץ I2C
- SDA לפין 8, SCL לפין 9

- ל-VCC 3.3V,GND לאරקה

2. סריקת כתובות:

- הרכזו את קוד הסריקה ובדקו אילו כתובות מזוהות
- רשמו את הכתובות של כל חישון

3. קריית נתונים:

- הרכזו כל דוגמת קוד בנפרד
- בדקו את התאמת הנתונים לתנאי הסביבה

4. שילוב שני החישונים:

- נסו לקרוא מכל החישונים באותו קוד
- הציגו את כל הנתונים (טמפרטורה, לחץ, לחות) יחד

נספח א' - בדיקת הספריות הזמינות לתוכנות ב- MicroPython תחת בקר ESP32

קישורים:

<https://docs.micropython.org/en/latest/library/index.html>

להלן הוראה:

```
help('modules')
```

להלן הפלט:

```
>>> help('modules')
__main__          bluetooth      heapq           select
 asyncio          btree         initsetup       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     framebuffer  random          webrepl
 asyncio/stream   gc           re               webrepl_setup
 binascii          hashlib      requests/_init_ websocket
 Plus any modules on the filesystem
>>> |
```

נספח ב' - יסודות בתכנות אסינכרוני ב-Python

קישורים:

<https://docs.python.org/3/library/asyncio-task.html>

פרק זה מתיאח לתוכנות אסינכרוני ב- Python (על מחשב PC או Google Colab)

להלן קישור למחברת Google Colab ה כוללת את כל הקודים שבנספח זה:

<https://colab.research.google.com/drive/1BpKcYEd82z-l-3oxFMUIrb0jD3mE83mU?usp=sharing>

מטרות:

1. להבין את המושגים הבסיסיים של תוכנות אסינכרוני.
2. ללמוד כיצד להשתמש ב-`async` ו-`await` ב- Python.
3. להכיר את המחלקה `Task`.`asynchronous` וכיצד להשתמש בה.
4. לישם תוכנות אסינכרוני כדי לשפר את ביצועי התוכנית.

לפניהם נראה מהו תכנון אסינכרוני, נבון תחילת מהו תכנון סינכרוני באמצעות התרגיל הבא:

תרגיל

פתחו קובץ Python על מחשב PC או מחברת ב- Google Colab ו כתבו בו את הקוד הבא:

```
def long_process():
    print("Long Process Started")
    x = 0
    for i in range(10000):
        for j in range(10000):
            x += i
    print(x)
    print("Long Process Completed")

def short_process():
    print("Short Process Started")
    for i in range(10):
        print(i, end=" ")
    print("\nShort Process Completed")

long_process()
short_process()
```

1. הסבירו מה התוכנית עשויה.
2. מה פلت התוכנית.
3. כמה זמן היה צריך להמתין עד שהפעולה `short_process` תופעל.
4. בעקבות הרצת התוכנית הסבירו במילים שלכם מה זה תכנון סינכרוני.

תרגיל

הריצו עכשו את קוד הבא וענו על השאלות הבאות:

```
import asyncio

async def long_process():
    print("Long Process Started")
    x = 0
    for i in range(10000):
        for j in range(10000):
            x += i
        if i % 100 == 0:
            await asyncio.sleep(0)
    print(x)
    print("Long Process Completed")

async def short_process():
    print("Short Process Started")
    for i in range(10):
        print(i, end=" ")
        await asyncio.sleep(0) # משחרר שליטה ל-event loop
    print("\nShort Process Completed")

async def main():
    await asyncio.gather(long_process(), short_process())

# Run the main function in the event loop
asyncio.run(main())

# If you are using Jupyter Notebook or IPython, you can use the following
# code instead:
#await main()
```

- .1 מה פלט התוכנית?
- .2 הסבירו מה התוכנית עשו.
- .3 מה עשו הפעולות gather במחלקה `asyncio`?
- .4 כמה זמן היה צריך להמתין עד שהפעולה `Short_Process` תופעל?
- .5 בעקבות הריצת התוכנית הסבירו במילים שלכם מה זה תכונות אסינכרוני.
- .6 מחקו את ההוראה `await asyncio.sleep(0)` והריצו את התוכנית שוב. הסבירו את פלט התוכנית לאחר השינוי.

תרגיל

להלן 2 פועלות printRed ו- printGreen שרצות בצורה סינכרונית. הריצו את 2 הפעולות באופן **אסינכריוני**. בצעו השוואה בין שני הפליטים (סינכרונית ואסינכרונית) והסבירו כיצד נוצר שוני זה.

```
RED = '\033[91m' # RED color codes
GREEN = '\033[92m' # GREEN color codes
BLUE = '\033[94m' # BLUE color codes
RESET = '\033[0m'  #Resets the color back to default

# Simple example
print(RED , "Text in RED color!" , RESET)
print(GREEN , "Text in GREEN color!" , RESET)
print(BLUE , "Text in BLUE color!" , RESET)

def print_red():
    for i in range(1, 101):
        print(RED , i , RESET, end="")

def print_green():
    for i in range(1, 101):
        print(GREEN , i , RESET, end="")

def main():
    print_red()
    print_green()

main()
```

העברת פרמטר לפעולה המופעלת באופן אסינכרוני.

להלן דוגמה להעברת פרמטרים לפעולה אסינכרונית:

```
import asyncio

RED = '\033[91m' # RED color codes
GREEN = '\033[92m' # GREEN color codes
BLUE = '\033[94m' # BLUE color codes
RESET = '\033[0m'  #Resets the color back to default

async def print_color(color=RED):
    for i in range(1, 101):
        print(color , i , RESET, end="")
        await asyncio.sleep(0)

async def main():
    await asyncio.gather(print_color(RED) , print_color(GREEN) ,
print_color(BLUE))

#await main()
asyncio.run(main())
```

סיכום ביניים:

מה זה Event Loop?

ה-Loop (לולאתאירועים) הוא הלב של מערכת אסינכרונית.

תשוחט על מלצר בمساعدة – הוא לא מחכה ליד כל שולחן עד שהוא כל ייה מוכן. הוא עובר בין השולחנות, לוקח הזמן, מחלק מנתות, עובר להלאה, ומדי פעם חוזר לשולחנות לבדוק מה השתנה.

בօפן דומה:

- ה-**Event Loop** עובר בין משימות (CoRoutines).
 - כשהוא רואה שימושה "תקועה" (נניח מחכה לרשף), הוא משעה אותה וועבר למשימה הבאה.
 - ברגע שהמשימה "התעוררה", הוא ממשיך אותה מנקודת נקודה.

חשוב להבין שכל הקוד שלנו רץ בתוך תהליך יחיד (Single Thread) – אבל בזכות ה-loop אנחנו מרחיכים סוג של "ריבוי משימות", בלי באמת לרטז במקביל.

מה זה Coroutine ?

coroutine היא פונקציה שניית להשנות אותה ולהציג שליטה ל-loop event לשבור את המקבלה.

אר מזהים coroutine בפייתן?

- מגדירים עם `async def`
 - מפעילים עם `await`

למה זה שימושי?

אפשר לכתוב קוד קריָא כמו סינכרוני, אבל שיתנהג כמו אסינכרוני – ימתין מבל' לחסום את כל התוכנית.

כאשר אנו מזמינים את הפעולה `await asyncio.sleep` הולך לטפל במשהו אחר במקומ לחרכות שנייה.

להלן ריכוז אפשרויות נוספות לביצוע בעותות אסינכרוניות ב- Python:

טכניתה	אייפה היא ריצה	ניהול ע"י	במקביל אמיית?	שיטוף זיכרון	מתאים ל...
Coroutine	Thread אחד ייחד	Event Loop	✗ (מדומה)	✓	O/I – רשות, קבצים, זמן אמת
Thread	מספר Threads	מערכת הפעלה	חולקת (GIL)	✓	קוד חום פשוט, UI, רקע
Process	תהליך נפרד לגמרי	מערכת הפעלה	✓ (אמית) ✓ (אין שיטוף)		чисוב כבד מאוד (CPU-bound)
"ישום נפרד"	מערכת נפרדת לגמרי	אתה (או Docker)	✓	✗	שירותים עצמאיים, מיקוד-שירותים

קיבלת פרמטר מפעולה המופעלת באופן אסינכרוני.

להלן דוגמת קוד לאופן שבו פועלה אסינכרונית מוחזירה ערכיים:

```
import asyncio

async def calculate_sum(a, b):
    print("...מח�יל חישוב")
    await asyncio.sleep(1) # מדמה פעולה ארוכה
    result = a + b
    print("!סיום חישוב")
    return result

async def main():
    # קבלת הערך המוחדר באמצעות await
    sum_result = await calculate_sum(3, 4)
    print("התוצאה היא:", sum_result)

await main()
#asyncio.run(main())
```

לסיכום:

async משמשת להכרזה על פעולה אסינכרונית. פעולה אסינכרונית יכולה לבצע קוד מבלי לחסום את התהליך (thread) שזמן את הפעולה. (כי בפעולה לריגילה התהליך (thread) שזמן את פעולה נחסם עד

אשר הפעולה מסתיימת). יצירת פעולה ה כוללת הצהרה בשם **async** מאפשרת לפעולה להשתמש ב- **await** כדי להמתין באופן אסינכרי להשלמת פעולה אסינכרכיות אחרות.

await משמשת לזמן פעולה, שבדרכ כל צו שדורשת זמן, בעוד פעולה אסינכרכית אחרת. זאת במטרה לציין שיש להמתין להשלמת ביצוע הפעולה. זה מאפשר לתוכנית להמשיך לבצע משימות אחרות תוך המתנה לסיום הפעולה המיוصلة.

נספח ג' - יסודות בתכנות אסינכרוני ב-MicroPython

פרק זה מתייחס לתוכנות אסינכרוני ב- Python MicroPython (תוך שימוש בbbc ESP32)

קישורים:

<https://docs.python.org/3/library/asyncio-task.html>

למה אנחנו צריכים תוכנות אסינכרוני:

1. לעיתים קרובות במערכות משובצות מחשב יש צורך לבצע מספר משימות במקביל (לדוגמה: קריית חישנים, הפעלת מנועים, תקשורת).
2. ללא תוכנות אסינכרוני, כל פעולה תחסום את הביצוע של פעולות אחרות.
3. תוכנות אסינכרוני מאפשרות לנו לבצע משימות במקביל בצורה יעילה.

הfonקציות העיקריות ב- asyncio של MicroPython :

create_task()	يוצר مهمة חדשה שתורץ במקביל למשימות אחרות
sleep()	משהה את הביצוע למספר שניות מבלי לחסום משימות אחרות
sleep_ms()	כמו sleep אבל במילישניות
gather()	אפשר להריץ מספר משימות במקביל ולחכotta שכון יסתינו
wait_for()	מריץ משימה עם timeout מוגדר

להלן דוגמת קוד

```
import asyncio
from machine import Pin
import time

# הגדרת פינים לדוגמה
led1 = Pin(2, Pin.OUT)
led2 = Pin(4, Pin.OUT)
button = Pin(5, Pin.IN, Pin.PULL_UP)

async def blink_led(led, interval):
    while True:
```

```

        led.value(not led.value())
        await asyncio.sleep_ms(interval)

async def check_button():
    while True:
        if not button.value(): # כפתור נלחץ
            print("Button pressed!")
            await asyncio.sleep_ms(200) # למניעת ריטוסט
        await asyncio.sleep_ms(50)

async def read_sensor():
    """הדמיה של קריית חיישן"""
    while True:
        # הדמיה קריית חיישן שלוקחת זמן
        print("Reading sensor...")
        await asyncio.sleep_ms(1000)
        print("Sensor value: ", time.ticks_ms() % 100)

async def main():
    # יצירה משימות שרצות במקביל
    task1 = asyncio.create_task(blink_led(led1, 500))
    task2 = asyncio.create_task(blink_led(led2, 1000))
    task3 = asyncio.create_task(check_button())
    task4 = asyncio.create_task(read_sensor())

    # הרצת כל המשימות במקביל #
    await asyncio.gather(task1, task2, task3, task4)

# הפעלת הלולאה האсинכראונית #
try:
    asyncio.run(main())
except KeyboardInterrupt:
    print("Program stopped by user")

```

בדוגמה זו אנחנו רואים מספר יתרונות של תכונות אסינכרוני:

1. **ביצוע מקביל**: אנחנו מבצעים 4 משימות במקביל:

- הבהיר LED ראשוני

- הבהיר LED שני במהירות שונה

- בדיקת לחיצה על כפתור

- קריית חיישן

2. **יעילות**: כל משימה מתבצעת בזמן המתאים מבלתי חוסום משימות אחרות

3. **זמן פשוט**: שימוש ב-sleep_ms(`await asyncio.sleep_ms(200)`) מאפשר זמן פשוט ומדויק

4. **קוד נקי**: הקוד מארגן בפונקציות לוגיות נפרדות שקל להבין ולתחזק

מבנה לכתיבת תוכנית המבוססת על תכונות אסינכרוני

להלן דוגמת קוד לתוכנית הכוללת 3 תתי תוכניות שכל אחת מהם מימושת פעולה נפרדת, כל תוכנית מבצעת קוד שונה בזמן תגובה שונים אבל **כלן רצות יחד!!**

```
import asyncio

async def myTask1():
    while True:
        #
        #
        #
        print('I am myTask1 ')
        await asyncio.sleep_ms(2000)

async def myTask2():
    while True:
        #
        #
        #
        print('I am myTask2 ')
        await asyncio.sleep_ms(1000)

async def myTask3():
    while True:
        #
        #
        #
        print('I am myTask3 ')
        await asyncio.sleep_ms(500)

#Run all tasks at the same time
async def main():
    t1 = asyncio.create_task(myTask1())
    t2 = asyncio.create_task(myTask2())
    t3 = asyncio.create_task(myTask3())
    await asyncio.gather(t1, t2, t3)

#Running the main program
try:
    asyncio.run(main())
except KeyboardInterrupt:
    print("Program stopped by user")
```

נקבל את הפלט הבא:

```
28     async def main():
29         t1 = asyncio.create_task(myTask1())
30         t2 = asyncio.create_task(myTask2())
31         await asyncio.gather(t1, t2)
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
```

להלן גרסה נוספת התוכנית הכוללת טיפול בשגיאות זמן ריצה:

```
import asyncio
async def myTask1():
    while True:
        try:
            #
            #
            #
            print('I am myTask1 ')
        except asyncio.CancelledError:
            print("Peripheral task canceled")
        except Exception as e:
            print("Error in ConnectionTask:", e)
        finally:
            await asyncio.sleep_ms(2000)

async def myTask2():
    while True:
        try:
            #
            #
            #
            print('I am myTask2 ')
        except asyncio.CancelledError:
            print("Peripheral task canceled")
```

```

        except Exception as e:
            print("Error in ConnectionTask:", e)
    finally:
        await asyncio.sleep_ms(1000)

async def myTask3():
    while True:
        try:
            #
            #
            #
            print('I am myTask3 ')
        except asyncio.CancelledError:
            print("Peripheral task canceled")
        except Exception as e:
            print("Error in ConnectionTask:", e)
    finally:
        await asyncio.sleep_ms(500)

#Run all tasks at the same time
async def main():
    t1 = asyncio.create_task(myTask1())
    t2 = asyncio.create_task(myTask2())
    t3 = asyncio.create_task(myTask3())
    await asyncio.gather(t1, t2, t3)

#Running the main program
try:
    asyncio.run(main())
except KeyboardInterrupt:
    print("Program stopped by user")

```

להלן מספר דוגמאות קצרות ליישומים שונים בפעולות אסינכרוניות

הפעלת פועלה בודדת כל שנייה

```

import uasyncio as asyncio

async def myTask():
    count = 0
    while True:
        count += 1
        print(count)
        await asyncio.sleep(1)

loop = asyncio.get_event_loop()
loop.create_task(myTask())
loop.run_forever()

```

uasyncio מספק לנו קבוצה של מחלקות ופעולות להפעלת משימות אסינכרוניות ותיאום ביצוען. "נעילה" היא אחת המחלקות המספקות על ידי uasyncio, שהוא אובייקט Ai הכללה הדדי המשמש לסנסר גישה למשאים

משותפים. הנעילה יכולה להיות באחד משני מצבים: "נעול" או "לא נעול". כאשר מנעול ננעול, כל קוד שמנסה לגשת אל המשאב הנעול ייחסם עד לשחרור המנעול. זה יכול להיות שימושי כדי להבטיח שימושה אחת בלבד יש גישה למשאב משותף בכל פעם, על מנת למנוע בעיות סyncron ותיאום בין משאים. להלן דוגמה להשתמש ב- Lock :MicroPython

```
import uasyncio as asyncio

# Create a lock
lock = asyncio.Lock()

async def myTask1():
    # Acquire the lock
    await lock.acquire()
    try:
        # Access shared resource
        print("myTask 1: acquired lock")
    finally:
        # Release the lock
        lock.release()

async def myTask2():
    await lock.acquire()
    try:
        print("myTask 2: acquired lock")
    finally:
        lock.release()

# Create an asyncio event loop
loop = asyncio.get_event_loop()
# Add tasks to the event loop
loop.create_task(myTask1())
loop.create_task(myTask2())
# Run the main loop
loop.run_forever()
```

בדוגמה זו, myTask1 ו-myTask2 הם משימות אסינכרוניות ששותפות ל洛克. מנסה לרכוש את הנעילה לפני גישה למשאב משותף. כאשר אחת המשימות רוכשת את המנעול, המשימה השנייה תיחסם עד לשחרור המנעול. זה מבטיח שרק משימה אחת יכולה לגשת למשאב המשותף בכל פעם.

להלן דוגמה נוספת לשימוש ב- Lock

```
import uasyncio as asyncio

async def task(i, lock):
    while 1:
        await lock.acquire()
        print("Acquired lock in task", i)
        await asyncio.sleep(0.5)
        lock.release()
```

```

async def killer():
    await asyncio.sleep(10)

loop = asyncio.get_event_loop()

lock = asyncio.Lock() # Global Lock instance
loop.create_task(task(1, lock))
loop.create_task(task(2, lock))
loop.create_task(task(3, lock))

loop.run_until_complete(killer()) # Run for 10s

```

להלן גרסה של שיטות של דוגמת קוד לתוכנית הכוללת 3 תתי תוכניות שכל אחת מהם מימושת פעולה נפרדת, כל תוכנית מבצעת קוד שונה בזמן תגובה שונים אך הפעם אין רצות יחד אותה אחת ממתינה לסיום של השניה כי יש חשש לשימוש באותם משאבים.

```

import asyncio

async def myTask1(lock):
    while True:
        try:
            await lock.acquire()
            #
            #
            #
            print('I am myTask1 ')
        except asyncio.CancelledError:
            print("Peripheral task canceled")
        except Exception as e:
            print("Error in ConnectionTask:", e)
        finally:
            await asyncio.sleep_ms(100)
            lock.release()

async def myTask2(lock):
    while True:
        try:
            await lock.acquire()
            #
            #
            #
            print('I am myTask2 ')
        except asyncio.CancelledError:
            print("Peripheral task canceled")
        except Exception as e:
            print("Error in ConnectionTask:", e)
        finally:
            await asyncio.sleep_ms(1000)
            lock.release()

async def myTask3(lock):
    while True:

```

```

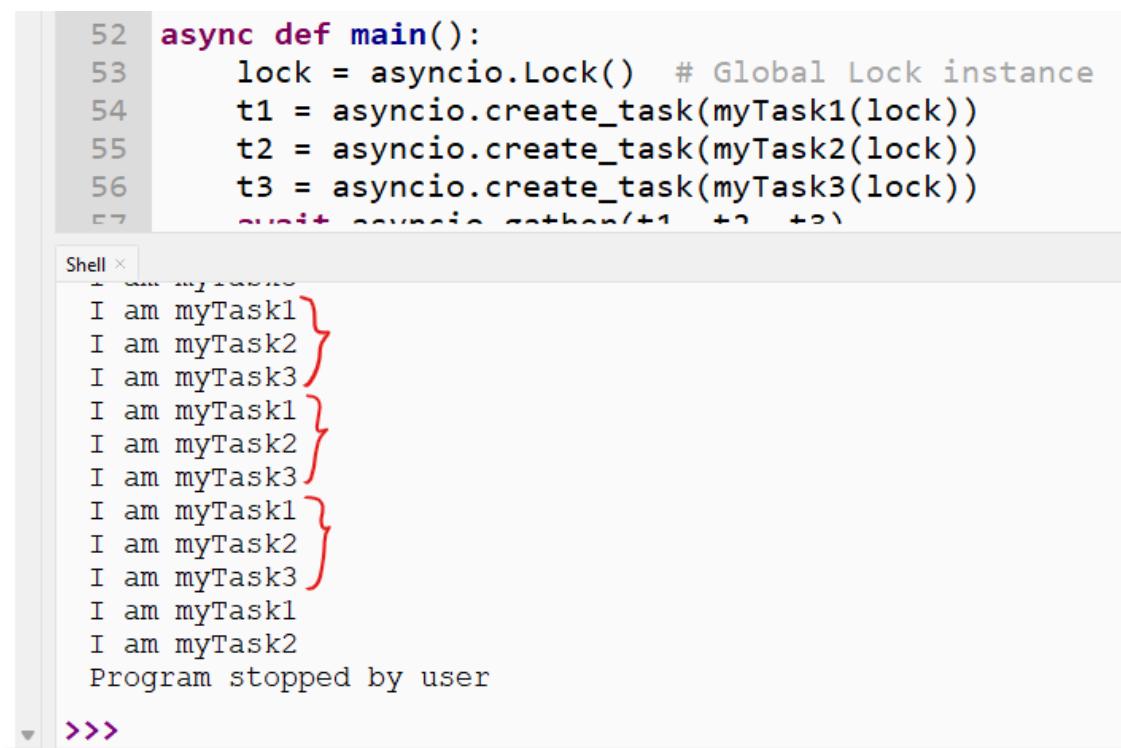
try:
    await lock.acquire()
    #
    #
    #
    print('I am myTask3 ')
except asyncio.CancelledError:
    print("Peripheral task canceled")
except Exception as e:
    print("Error in ConnectionTask:", e)
finally:
    await asyncio.sleep_ms(500)
    lock.release()

#Run all tasks at the same time
async def main():
    lock = asyncio.Lock() # Main Lock instance
    t1 = asyncio.create_task(myTask1(lock))
    t2 = asyncio.create_task(myTask2(lock))
    t3 = asyncio.create_task(myTask3(lock))
    await asyncio.gather(t1, t2, t3)

#Running the main program
try:
    asyncio.run(main())
except KeyboardInterrupt:
    print("Program stopped by user")

```

ניתן לראות שהפעולות כוללות זמן תגובה שונים אך בפועל אחת ממתינה לשניה:



```

52 52  async def main():
53 53      lock = asyncio.Lock() # Global Lock instance
54 54      t1 = asyncio.create_task(myTask1(lock))
55 55      t2 = asyncio.create_task(myTask2(lock))
56 56      t3 = asyncio.create_task(myTask3(lock))
      57  await asyncio.gather(+1 +2 +3

```

Shell >>>

```

I am myTask1 }
I am myTask2 }
I am myTask3 }
I am myTask1 }
I am myTask2 }
I am myTask3 }
I am myTask1 }
I am myTask2 }
I am myTask3 }
I am myTask1 }
I am myTask2 }
Program stopped by user

```

דוגמה מעשית לשימוש בקוד הנועל משאב תוך שימוש ב-`Lock`.

כאשר יש משאב מסוותף (כמו קובץ או חישון), ושתי MISIMOT או יותר עלולות לגשת אליו בו-זמנית, שימוש ב-`Lock` מבטיח שרק משימה אחת תיגש אליו בכל רגע נתון. נדגים זאת על ידי המצב הבא:

- יש 2 חישונים (חישון טמף' לדוגמה, חישון א/or).
- כל חישון נמדד בתדרות שונה.
- כל נתון נכתב לקובץ (משותף).
- השתמש ב-`Lock` כדי למנוע כתיבת סימולטנית לאותו קובץ.

להלן דוגמת קוד:

```
import uasyncio as asyncio
from machine import ADC, Pin
import time

temp_sensor = ADC(Pin(34))
light_sensor = ADC(Pin(35))

async def write_to_file(lock, filename, data):
    await lock.acquire()
    try:
        with open(filename, 'a') as f:
            f.write(data + '\n')
    finally:
        lock.release()

async def sample_temperature(lock):
    while True:
        value = temp_sensor.read()
        timestamp = time.time()
        data = f'{timestamp}, temp, {value}'
        print(data)
        await write_to_file(lock, 'sensor_data.txt', data)
        await asyncio.sleep(2)

async def sample_light(lock):
    while True:
        value = light_sensor.read()
        timestamp = time.time()
        data = f'{timestamp}, light, {value}'
        print(data)
        await write_to_file(lock, 'sensor_data.txt', data)
        await asyncio.sleep(5)

async def main():
    lock = asyncio.Lock()
    t1 = asyncio.create_task(sample_temperature(lock))
    t2 = asyncio.create_task(sample_light(lock))
    await asyncio.gather(t1, t2)

try:
```

```

    asyncio.run(main())
except KeyboardInterrupt:
    print("Program stopped by user")

```

מה עושה הקוד:

- כל מ시ימה מודדת חישון בתדרות שונה (sleep שונה).
- כל פעם שנכתב משהו לקובץ – מatabase acquire ל-Lock.
- אם מושימה אחרת רוצה לנכט לקובץ – היא תמתין לשחרור ה-Lock. כך נמנעת כתיבת בו-זמןית שעלולה לגרום לשגיאות או קובץ פגום.

שאלה:

איך הבקר יודע מה לנעול? איך הוא מבין שצריך לנעול את גישה הכתיבה לקובץ?

תשובה:

הבקר לא יודע בלבד מה "צורך לנעול" – אנחנו כמפתחים צרכים להחליט מהו המשאב המשותף, ולזוזה שכל גישה אליו נעשית תחת נעליה (Lock). ה-Lock הוא אובייקט לוגי, לא קשר ישירות לקובץ או לחישון. אנחנו יוצרים אותו כך:

lock = asyncio.Lock()

ואז עוטפים אותו את הקוד שדורש "בלעדיות".

שאלה:

מה קורה כאשר תהליך מנסה לנעול (acquire) את הגישה לקובץ אבל הקובץ כבר "נעול"?

תשובה:

כאשר מושימה אחת מבצעת וה-Lock כבר תפום, קלומר מושימה אחרת כבר ביצעה acquire ולא שחררה (release) אז המושימה ממתינה (await) עד שה-Lock ישוחרר. היא לא נמשכת הלאה, אבל גם לא קורסת או נתקעת – היא פשוט "ישנה" ברקע ומחכה לתורה.

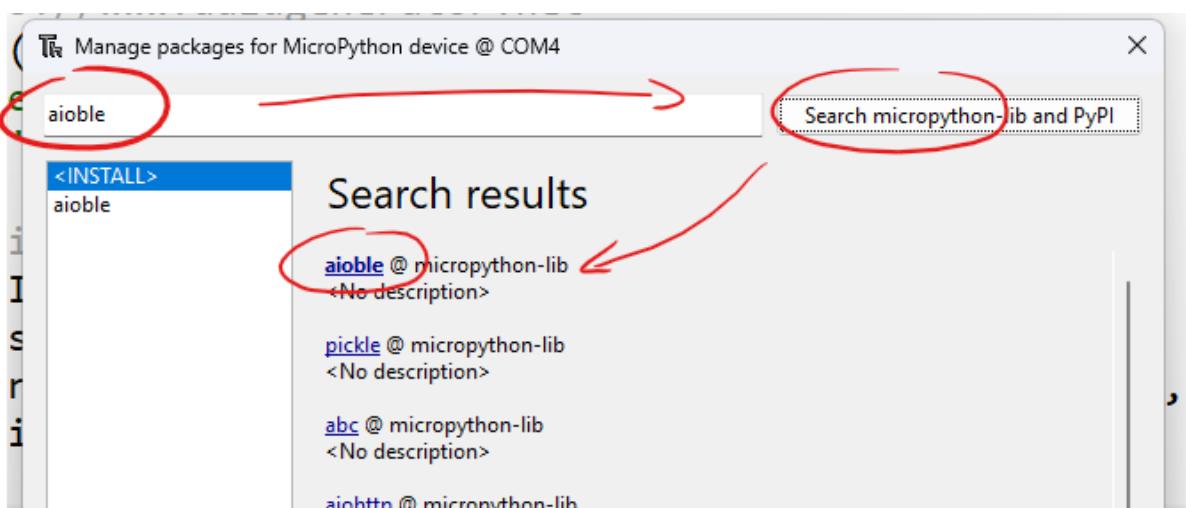
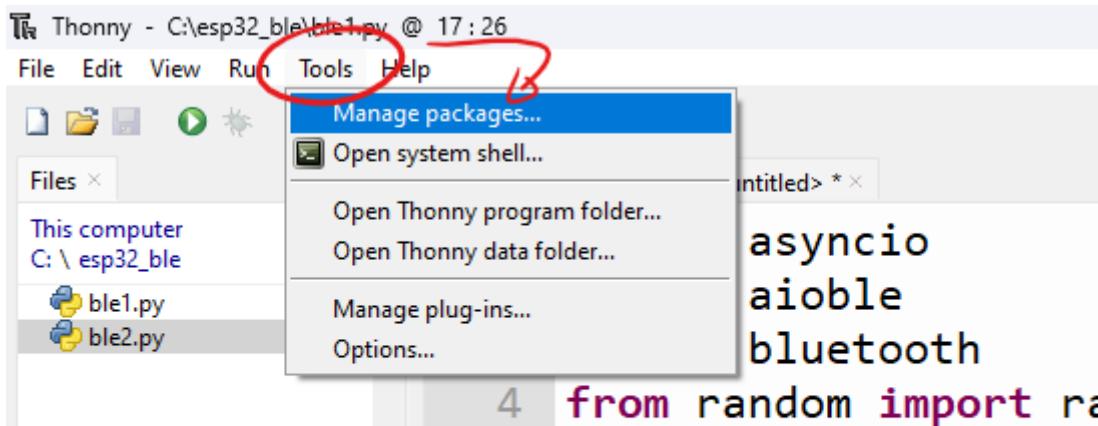
לסיכום:

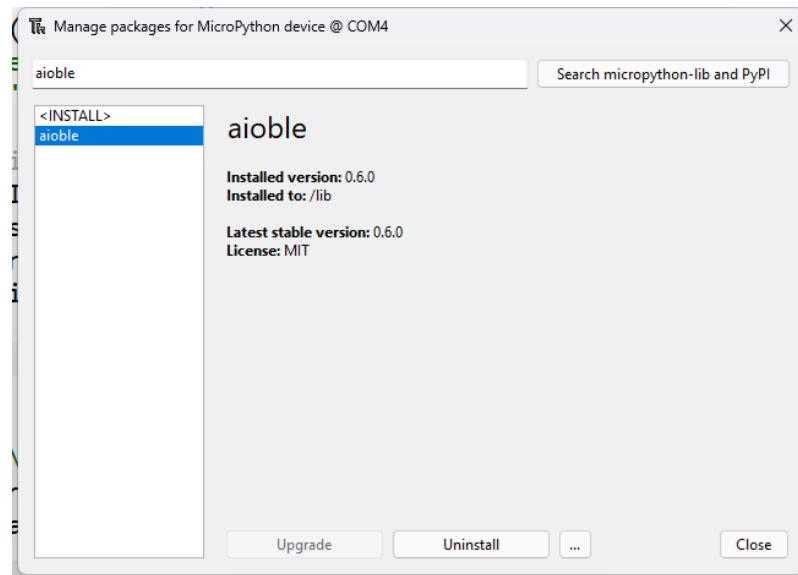
מה קורה בפועל	מצב
מה קורה בפועל	מצב

נספח ד' - יבוא ספרייה קוד ייעודיות ל- MicroPython

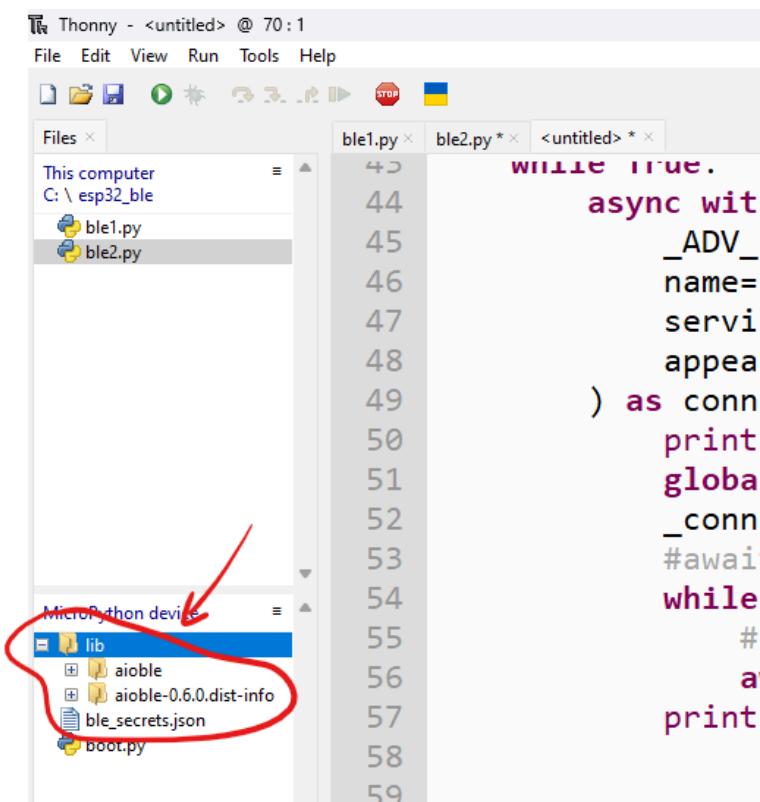
שימוש לבן את החבילה אנו מתקנים ישירות על הבקר, על כן יש לחבר אותו למחשב לפני תחילת תהליך ההתקנה.

כדי להתקין את חבילת הקוד aioble נפתח את סביבת העבודה Thonny וナルץ על tools→Manage packages

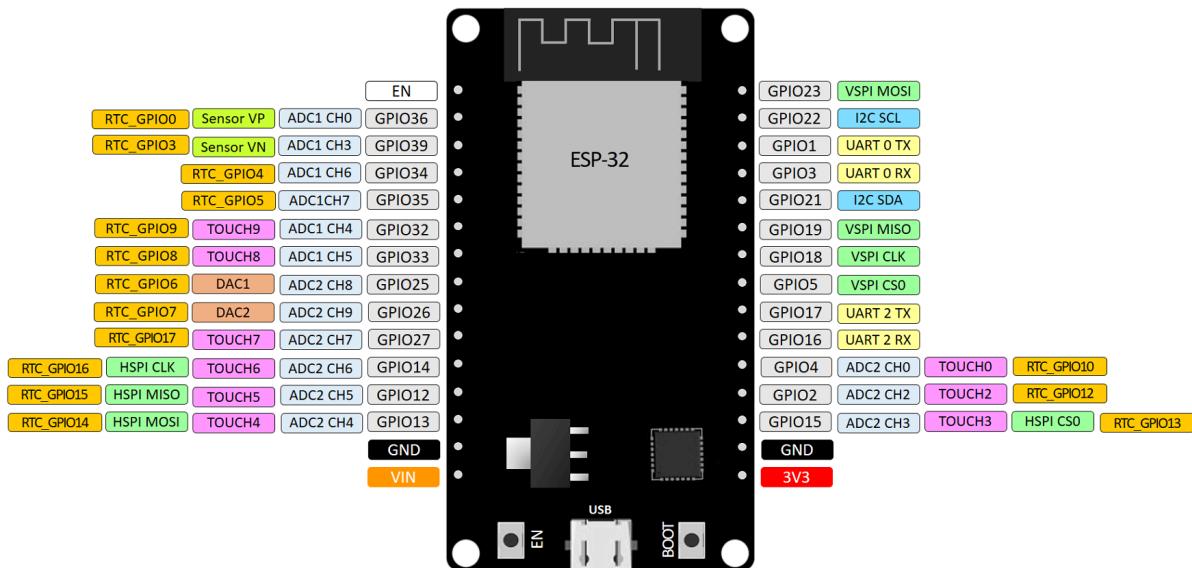




שימוש לב של אחר ההתקנה תפתח בברק ספריה חדשה בשם lib ה כוללת את הקבצים של חבילת התוכנה:



נספח ה' - מיפוי הדקי בקר ESP32



*קיים בשוק מוצר ESP32 שבהמ הדקי הרכיב ממופים באופן שונה.

מיפוי הדקי הקלט/פלט של בקר ESP32 :

Notes	Output	Input	GPIO	Notes	Output	Input	GPIO
	OK	OK	16	outputs PWM	OK	pulled up	0
	OK	OK	17	debug output at boot	OK	TX pin	1
	OK	OK	18	on-board LED	OK	OK	2
	OK	OK	19	HIGH at boot	RX pin	OK	3
	OK	OK	21		OK	OK	4
	OK	OK	22	outputs PWM	OK	OK	5
	OK	OK	23	SPI flash	x	x	6
	OK	OK	25	SPI flash	x	x	7
	OK	OK	26	SPI flash	x	x	8
	OK	OK	27	SPI flash	x	x	9
	OK	OK	32	SPI flash	x	x	10
	OK	OK	33	SPI flash	x	x	11
input only		OK	34	boot fail if pulled high	OK	OK	12
input only		OK	35		OK	OK	13
input only		OK	36	outputs PWM	OK	OK	14
input only		OK	39	On board LED -PWM	OK	OK	15

*קיים בשוק מוצר ESP32 שבהמ הדקי הרכיב ממופים באופן שונה.

נספח ו' - עדכון קושחה לבר ESP32

מקור:

<https://www.youtube.com/watch?v=4kiNU-dNcf0>

נתקין תוכנה המאפשרת לעדכן קושחה מהכתובת הבאה:

<https://www.espressif.com/en/support/download/other-tools>

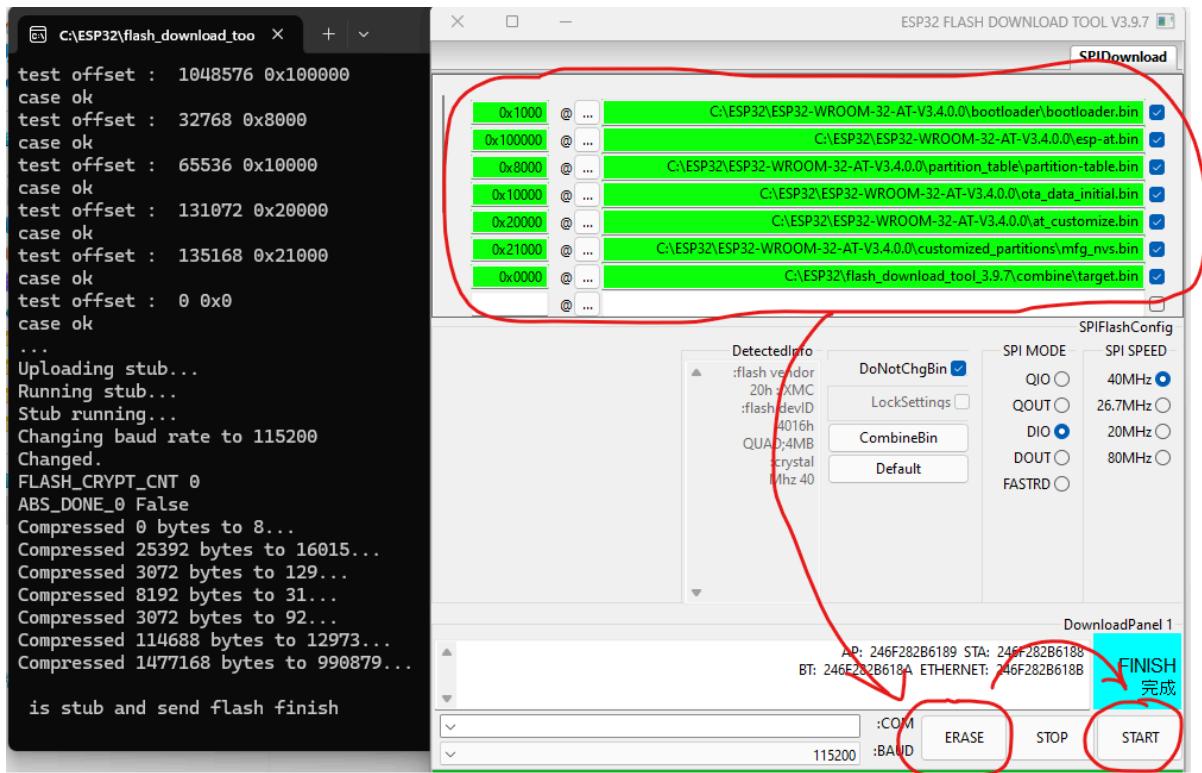
The screenshot shows the Espressif Download page. At the top, there's a navigation bar with links for Hardware, SDKs, Cloud, Solutions, Support (which is underlined), Ecosystem, Company, Contact, and language selection (EN / 中文). Below the navigation is a search bar with the placeholder "Search keywords". A sidebar on the left has sections for All, SDKs & Demos, Apps, Tools (which is selected and highlighted in red), and ESP-AT. The main content area has a heading "Download" with a background image of binary code and a blue toolbox icon. Below it, a table lists "Flash Download Tools". The table has columns for Title, Platform, Version, Release Date, and Download. One row is shown: "+ Flash Download Tools" (with a red circle around it) for Windows PC, V3.9.6, 2024.06.07, with a download link (also circled in red).

נוריד את הגרסה העדכנית ביותר לקושחה שבאתר הבא:

https://docs.espressif.com/projects/esp-at/en/latest/esp32/AT_Binary_Lists/esp_at_binaries.html

The screenshot shows the Espressif AT Binary Lists page for the Released Firmware. The left sidebar includes a dropdown for "Technology" set to "ESP32" (circled in red), a "Search docs" input field, and a "Get Started" button. Under "AT Binary Lists", there are sections for ESP32 AT Released Firmware (including WROOM-32, MINI-1, WROVER-32, PICO, SOLO series), AT Command Set, AT Command Examples, ESP-AT Versions, Compile and Develop, Customized AT Commands and Firmware, FAQ, and Index of Abbreviations. The main content area has a heading "Released Firmware" and a note about using the latest version. It lists the ESP32 series modules. A "Note" section provides instructions for generating new firmware if no released version exists. The "ESP32-WROOM-32 Series" section shows three firmware files: v3.4.0.0 ESP32-WROOM-32-AT-V3.4.0.0.zip (Recommended), v3.2.0.0 ESP32-WROOM-32-AT-V3.2.0.0.zip, and v2.4.0.0 ESP32-WROOM-32-AT-V2.4.0.0.zip. The first file is circled in red.

נפתח את התוכנה ונקבע בה את הפרמטרים הבאים:



נחבר את ה-ESP32 למחשב ונלחץ על ERASE לאחר קבלת הודעה FINISH נלחץ על START כדי לזרוב את התוכנה העדכנית.

קישורים למקורות מידע נוספים

מاجر קישורים למקורות מידע ועוד ב- [Micropython](#)

<https://awesome-micropython.com/>

גרסת ONLINE לסביבת פיתוח IDE עבור [Micropython](#)

<https://viper-ide.org/>

...

תנאי השימוש

תנאי השימוש במסמך זה האם לפי הסטנדרט הבא:

You are free:

- to Share** – to copy, distribute and transmit the material
- to Remix** – to adapt the material

Under the following conditions:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.