

התקשורת הטורית ב ESP32 ו Hardware Serial של סביבת ARDUINO IDE

א. במה נעסוק בפרק זה ?

בפרק כאן נדבר על תקשורת UART שהיא Universal Asynchronous Receiver Transmitter - מקלט/משדר אסינכרוני אוניברסלי ונביא דוגמה לתקשורת טורית בין שני כרטיסי ESP32 באמצעות ספריית חומרת UART של Arduino IDE. ב-ESP32 יש חיבור USB למחשב לצורך תכנות ("צריבה") של זיכרון התוכנית וכדי לאתר באגים בעזרת המוניטור (מסך) טורי של סביבת העבודה IDE של הארדואינו.

השימוש בתקשורת ה-UART נפוץ מאוד עבור הרבה חיישנים ומערכות כמו בלוטות, GPS, רכיבי קול ועוד. לפעמים נדרשת תקשורת בין שני כרטיסי ESP32 כדי לשתף עומס עבודה, מידע ולבצע משימות שונות. אנו נשתמש ב-Arduino IDE כדי לתכנת את התקשורת בין 2 הכרטיסים.

אנחנו נשתמש ב-Arduino IDE כדי לתכנת את כרטיסי הפיתוח ESP32 שלנו. לכן, אנחנו צריכים את הגרסה האחרונה של Arduino IDE ובנוסף, להתקין גם את תוסף (Plugin) עבור ESP32.

אם ב-IDE שלנו לא מותקן התוסף ניתן לבקר בקישורים הבאים שבהם הסבר על התקנת ספריית ESP32 ב-Arduino IDE :

<https://www.arikporat.com/wp-content/uploads/2023/01/introduction-to-esp32.pdf>

שבו הסבר בעברית על ההתקנה או הקישור הבא שבו הסבר באנגלית.

[How to Install ESP32 in Arduino IDE - step by step tutorial \(microcontrollerslab.com\)](https://microcontrollerslab.com/how-to-install-esp32-in-arduino-ide-step-by-step-tutorial/)

ב. מבוא ל UART

תקשורת טורית UART - היא אחד מפרוטוקולי התקשורת הפשוטים ביותר בין שני מכשירים. בתקשורת זו מעבירים נתונים בין מכשירים על ידי חיבור של שני חוטים בין המכשירים, האחד הוא קו השידור ואילו השני הוא קו הקליטה. הנתונים מועברים ביט אחרי ביט. היתרון העיקרי של פרוטוקול תקשורת זה הוא שאין צורך שלשני המכשירים תהיה שעון מסנכרן ולכן זו תקשורת אסינכרונית ! . לדוגמה, שני מיקרו-בקרים הפועלים בתדרי שעון שונים (לדוגמה ארדואינו שתדר השעון שלו הוא 16 מגה הרץ ו-ESP32 שתדר השעון שלו יכול להגיע ל 240 מגה הרץ , יכולים לתקשר זה עם זה בקלות באמצעות תקשורת טורית. עם זאת, קצב התקשורת הטורית עצמה, כלומר , קצב העברת הסיביות הנקרא BAUD חייב להיות מוגדר מראש ושווה בשני הרכיבים המתקשרים .

ג. שידור וקליטה של נתונים טוריים

ה-UART במשדר לוקח בתים של נתונים ומעביר את הביטים בצורה רציפה. המקלט בצד השני מרכיב מחדש את הביטים לביית שלם. שידור סדרתי של נתונים באמצעות חוט יחיד הוא למעשה חסכוני יותר מאשר שידור מקבילי דרך חוטים מרובים. זהו ייתרון גדול מבחינת מחיר אבל חיסרון מבחינת מהירות העברת הנתונים.

התקשורת בין שני התקני UART עשויה להיות חד-צדדית -SIMPLEX , דו-צדדית מלאה FULL DUPLEX או דו-צדדית למחצה - HALF DUPLEX. תקשורת SIMPLEX היא סוג של תקשורת חד-כיוונית שבה האות עובר מ-UART אחד למשנהו. אין לו הוראה ל-UART המקבל לשלוח אותות בחזרה. דופלקס מלא הוא כאשר שני המכשירים יכולים לשדר ולקבל תקשורת בו זמנית. חצי דופלקס הוא כאשר מכשירים מתחלפים לשדר ולקבל (כמו בתקשורת צבאית למשל).

ד. מבנה פרוטוקול UART

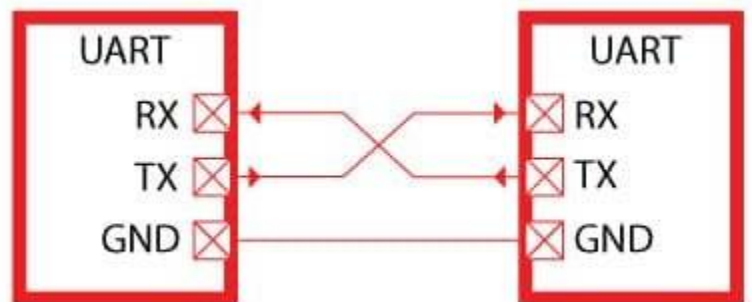
- **UART** מכיל מחולל שעון. זה מאפשר את הדגימה בפרק זמן קצר.
- הוא מכיל גם אוגרי הזזה של קלט ופלט.
- בקרת שידור וקליטה.
- לוגיקת בקרת קריאה או כתיבה.
- רכיבים אופציונליים אחרים של UART הם: חוצצים (BUFFERS) של שידור או קליטה (החוצץ ב ESP32 הוא של 128 בתים), זיכרון First In First Out - FIFO - ובקר גישה ישירה לזיכרון הנקרא – Direct Memory Access .

ה. טכנולוגיית UART

לפני זמן לא רב, לרכיבים כמו מקלדות, עכברים ומדפסות שהתחברו בתקשורת UART היו כבלים עבים וקונקטורים שהיו מוברגים אל המחשב. הכבלים המגושמים האלה הוחלפו ב- USB וניתן למצוא אותם ברכיבים כגון Raspberry Pi, Arduino ומיקרו-בקרים נפוצים אחרים. אנו יכולים להשתמש בו כדי לחבר מודולי Bluetooth ומודולי GPS. ל- UART פרוטוקול העברה שונה מפרוטוקולי תקשורת אחרים כגון SPI ו- I2C. זהו מעגל פיזי בתוך המיקרו-בקר. הוא יכול לתפקד גם כמעגל משולב עצמאי. יתרון משמעותי אחד של UART הוא שהוא מסתמך רק על שני חוטים כדי להעביר נתונים.

1. איך מחברים 2 רכיבים בתקשורת UART ?

האיור הבא מתאר חיבור של 2 רכיבים בתקשורת UART .



איור 1 : 2 רכיבים המחברים בתקשורת UART .

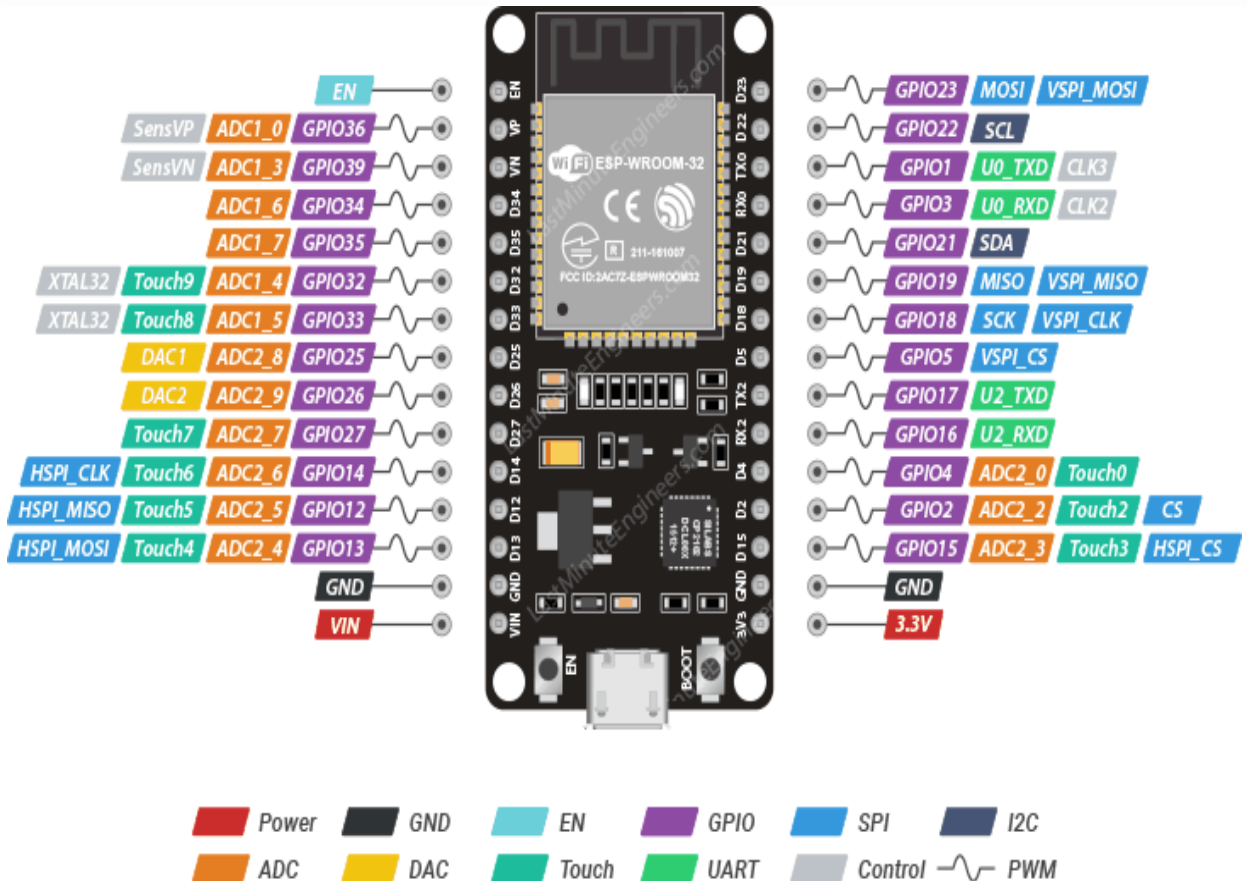
דרושים שני UARTs כדי לתקשר ישירות זה עם זה. בקצה אחד, ה-UART המשדר ממיר נתונים מקביליים ממעבד לצורה טורית, ואז משדר את הנתונים בצורה טורית ל-UART השני שמקבל את הנתונים הטוריים וממיר אותם בחזרה לנתונים מקביליים. לאחר מכן ניתן לגשת לנתונים אלה מהמכשיר המקבל. במקום להשתמש בשעון מסנכרן, הביט המשודר משתמש בתחילת שידור הבייט הנקרא ביט התחלה START ובסיום השידור בביט סיום - STOP עבור חבילות הנתונים. סיביות התחלה וסיום אלו מגדירים את ההתחלה והסוף של חבילות הנתונים. לכן UART מקבל יודע מתי מתחיל ביט של נתון ומתי מסתיים הבייט. ביט חדש יישלח שוב עם ביט START חדש ובסוף הבייט של הנתון יהיה ביט סיום כך הלאה.

ה- UART המקבל יזהה את סיבית ההתחלה ואז יתחיל לקרוא את הביטים. התדר המשמש לקריאת הסיביות הנכנסות ידוע בשם קצב השידור. קצב השידור הנקרא BAUD הוא מדד המשמש להראות את מהירות העברת הנתונים. היחידה שמתארת קצב שידור היא סיביות לשנייה (Bits Per Second - bps). על מנת שהעברת הנתונים תהיה מוצלחת, הן UART המשדר והן המקלט חייבים לפעול כמעט באותו קצב שידור עד הפרש של כ 10%. יש להגדיר את ה- UART המקבל והמשדר כך שידעו מכמה ביטים מורכב הנתון (יש כאלו שמשדרים 5 ביטים עד בדרך כלל ל 8 ביטים), האם יש ביט זוגיות - PARITY – והאם יש ביט עצירה אחד או יותר (אפשר ביט עצירה של 1.5 כלומר זמן הביט פי 1.5 מזמן ביט).

ז. הדקי ה UART ב ESP32

למיקרו ESP32 שלושה UART הנקראים ,UART0, UART1 ו-UART2 העובדים ברמת TTL של 3.3 וולט. שלושת הממשקים הטוריים הללו נתמכים בחומרה. לכל אחד מהם 4 קווים: TX, RX, RTS, ו-CTS. עם זאת, Arduino IDE משתמש רק בהדקים RX ו-TX.

האיור הבא מתאר את הדקי כרטיס ESP32 WROOM. הדקי התקשות הטורית UART מסומנים בירוק



איור 2: הדקי כרטיס ESP32 WROOM

באיור רואים את הדקי UART0 שמספרם GPIO1 ו-GPIO3 ואת הדקי UART2 שהם GPIO16 ו-GPIO17. בהרבה כרטיסים UART1 איננו מתחבר להדקים החיצוניים בכרטיס.

- UART0 משמש לתקשורת עם ESP32 לצורך תכנות ובמהלך איפוס/אתחול.

- UART1 אינו בשימוש וניתן להשתמש בו עבור הפרויקטים שלנו. עם זאת, לוחות מסוימים משתמשים ביציאה זו עבור גישה ל-SPI Flash ולכן לא מומלץ להשתמש בו. בהרבה לוחות UART1 איננו מחובר להדקי הכרטיס.
- UART2 אינו בשימוש וניתן להשתמש בו בפרויקטים שלנו.

כברירת מחדל, ניתן להשתמש רק ב-UART0 וב-UART2. כדי להשתמש ב-UART1, עלינו להגדיר מחדש את הפינים מכיוון שפיני ברירת המחל של UART1 שהם GPIO9 ו-GPIO10 מחוברים באופן פנימי לזיכרון SPI FLASH. כמו כן, בחלק מלוחות ESP32, הם נמצאים בהדקי הכרטיס ואיננו יכולים להשתמש ב-UART1 ישירות מבלי להקצות הדקים מחדש ב-Arduino IDE.

הטבלה הבאה מראה את הדקי RX ו-TX עבור כל אחת משלוש יציאות ה-UART הזמינות ב-ESP32.

| UART | RX | הדק השידור TX | הדק הקליטה RX | CTS | RTS | האם ניתן להשתמש ? |
|-------|--------|---------------|---------------|-------|--------|--|
| UART0 | GPIO3 | GPIO1 | | N/A | N/A | כן |
| UART | GPIO9 | GPIO10 | | GPIO6 | GPIO11 | כן, אך דורש הקצאה מחדש של הדקים 'קיי הג'וק לא מתחברים אל ההדקים 'צוניים בכרטיס). |
| UART | GPIO16 | GPIO17 | | GPIO8 | GPIO7 | כן |

טבלה 1 : ההדקים של 3 ה-UARTs ב-ESP32

ה. אתחול ה-UART של ESP32 בסביבת - Arduino IDE

נשתמש בספריית HardwareSerial בעת עבודה עם תקשורת UART ESP32 באמצעות יציאות UART1 או UART2. אז, ראשית נכלול את הספרייה בתוכנית בעזרת הפקודה :

```
#include <HardwareSerial.h>
```

לאחר מכן, נאתחל את הפורט הטורי הרצוי בעזרת יצירת אובייקט. אנו משתמשים בפקודות הבאות כדי להגדיר את יציאת ה-UART שבחרנו.

```
HardwareSerial (<מספר הפורט 1 או 2> < כל שם כלשהו שנבחר >);
```

לדוגמה אתחול של UART1 ושינינו את שמו ל myUart1 .

HardwareSerial myUart1 (1);

לאחר מכן בפונקציית ה `setup()` ניתן לאתחל אותו כך:

(מספר הדק ה TX , מספר הדק ה RX , פרוטוקול התקשורת , קצב התקשורת) .begin(השם שבחרנו בפקודה הקודמת

קצב התקשורת אומר כמה ביטים משודרים בשנייה (דוגמאות : 300 , 600 , 1200 , 2400 , 9600 , 14400 , 19200 , ועוד).

"פרוטוקול התקשורת" אומר מכמה ביטים מורכב ביית הנתון , האם יש ביט זוגיות וכמה ביטים יהיה ביט הסיום. לדוגמה :

SERIAL_5N1 אומר שהנתון של 5 ביטים ללא זוגיות ועם ביט סיום 1 . אם נרשום SERIAL_6E1 זה אומר שהנתון הוא של

6 ביטים , עם זוגיות זוגית – Even , עם ביט סיום 1 . אם נרשום SERIAL_7O2 אז הנתון הוא של 7 ביטים , הזוגיות היא

אי זוגית -Odd , ויש 2 ביטים של סיום.

מספר הדק RX יכול להיות כל מספר בין 0 ל 31 . מספר הדק TX יכול להיות כל הדק בין 0 ל 39 .

לדוגמה (עבור השם שנתנו מקודם – myUart1) :

`myUart1.begin(115200, SERIAL_8N1, 17, 18);`

במקום הדקים 17 ו 18 יכולנו לרשום הדקים אחרים ולהשתמש בהם.

לא חייבים לרשום את כל הנתונים שנמצאים בסוגריים בפקודה אבל אז עובדים עם הדקי ברירת המחדל ופרוטוקול ברירת

המחדל. אם נרשום רק את הפקודה : `myUart1.begin(115200);` זה אומר שמדובר ב UART1 , הוא עובד בקצב

תקשורת של 115200 ביטים בשנייה , עם נתון בגודל 8 ביטים, ללא זוגיות ועם ביט סיום אחד. הדקי השידור שלו הם 1 ו 3 .

דוגמה נוספת לתכנות ה UARTs :

// נשים את מספר הפורט בתוך הסוגריים

`Serial.begin()` // UART0 אם משתמשים ב

`HardwareSerial SerialPort(1)` // UART1 אם משתמשים ב

`HardwareSerial SerialPort(2)` // UART2 אם משתמשים ב

לאחר מכן, כמו בדוגמה קודמת , בתוך הפונקציה `setup()` , נאתחל את תקשורת UART באמצעות האובייקט `SerialPort`

בפקודה `Serial.begin()` ונציין ארבעה פרמטרים בתוכו באופן הבא:

`SerialPort.begin (BaudRate, SerialMode, RX_pin, TX_pin) ;`

הפרמטרים כוללים את קצב השידור `BaudRate` , הפרוטוקול הטורי – כמה ביטים יש בביית והאם יש ביט זוגיות וכמה ביטים של

סיום , מספר ההדק של קו RX וקו TX הנמצאים בשימוש. לדוגמה : אם נשתמש ב- UART2 , נאתחל אותו באופן הבא:

`#include <HardwareSerial.h>` // HardwareSerial שימוש בקובץ הכותר

`HardwareSerial SerialPort(2);` // UART2 יצירת אובייקט עבור

`void setup()`

{

// קצב תקשורת 115200 ביטים בשנייה בכל ביית 8 ביטים ללא זוגיות עם ביט סיום 1 הדק הקליטה 16 והדק השידור 17

`SerialPort.begin(115200, SERIAL_8N1, 16, 17);`

}
 אם נרצה להשתמש ב-UART1, כדאי לשים לב שהדק 9 והדק 10 שהם חלק מהדקי RX ו TX של UART1 מחוברים בתוך הרכיב לזיכרון ה FLASH שברכיב (ואינם מתחברים להדקים חיצוניים בכרטיס). לכן נצטרך להקצות מחדש את הפינים עבור UART1 לתקשורת טורית. למרבה המזל, לוח ESP32 מסוגל להשתמש כמעט בכל פניו GPIO עבור חיבורים טוריים בעזרת מערכת Cross Bar שיש לו. כאן הקצינו מחדש את GPIO4 כהדק RX ואת GPIO2 כהדק TX . הסבר נוסף על הקצאת הדקים ניתן למצוא בפרקים הבאים.

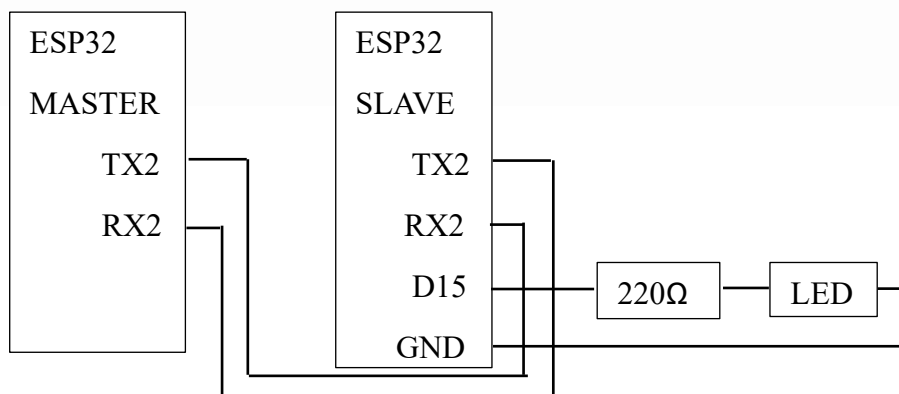
```
#include <HardwareSerial.h>
HardwareSerial SerialPort(1); // UART 1
void setup()
{
  SerialPort.begin(115200, SERIAL_8N1, 4, 2);
}
```

ט. תקשורת UART בין שני כרטיסי ESP32

ט.1 - הומרה

נראה דוגמה לתקשורת טורית שבה אדון ESP32 ישלח '1' או '0' לעבד ESP32. לאחר מכן העבד יקבל את הנתונים וישלוח בנורית LED המחוברת עם ההדק הדיגיטלי שלו. אנו נשתמש ב-UART2 כדי לתקשר בין שני הלוחות. נשתמש ברכיבים הבאים עבור פרויקט זה:

- 2 כרטיסי ESP32 .
- לד ונגד של 220 אוהם בטור לה.
- מגשרי קצר – jumpers .
- האיור הבא מתאר את החיבור :



איור 3 : חיבור UART בין 2 כרטיסי ESP32

נחבר את הדק TX2 שבכרטיס המאסטר עם הדק RX2 של לוח העבד. באופן דומה, נחבר את הדק RX2 של לוח מסטר עם הדק TX2 של לוח העבד. רגל האנודה של LED מחוברת להדק דיגיטלי 15 (בכרטיס העבד) דרך נגד הגבלת זרם של 220 אוהם. רגל הקתודה של הלד מחוברת לאדמה GND.

כמו כן נחבר את שני כרטיסי ה ESP32 לאדמה – GND משותפת.

ט.2 תוכנה

ט.2.א תוכנת ה MASTER:

```
#include <HardwareSerial.h>
HardwareSerial SerialPort(2); // UART2
void setup()
{
  SerialPort.begin(115200, SERIAL_8N1, 16, 17);
}
void loop()
{
  SerialPort.print(1);
  delay(5000);
  SerialPort.print(0);
  delay(5000);
}
SerialPort.begin(115200, SERIAL_8N1, 4, 2);
```

}

ט.2.א.1 – איך עובדת התוכנית במסטר ?

* ראשית נכלול את הספרייה הטורית של החומרה באמצעות השורה #include . שורה זו כוללת את ספריית HardwareSerial.h .
 * יוצרים אובייקט טורי של ספריית HardwareSerial בשם "SerialPort" . אנחנו מציינים את מספר פורט ה UART כפרמטר בתוכו. נשתמש ב- UART2 במקרה זה : HardwareSerial SerialPort(2); .
 * בתוך הפונקציה ה setup() נפתח את התקשורת הטורית של יציאה UART2 בעזרת הפקודה :

```
SerialPort.begin (BaudRate, SerialMode, RX_pin, TX_pin).
```

אנחנו עובדים בקצב תקשורת 115200 ביטים בשנייה, 8 ביטים ללא ביט זוגיות ועם ביט סיום 1 . ההדק 16 הוא קליטה ו 17 לשידור.

```
SerialPort.begin(115200, SERIAL_8N1, 16, 17);
```

* בפונקציית ה loop() נשלח בלולאה אין סופית את התו '1' (ערך אסקי 0x31), נמתין 5 שניות ואז נשלח את התו '0' (ערך אסקי 0x30) ונמתין 5 שניות וחוזר חלילה.

ט.2.ב התוכנה ב SLAVE

התוכנית בעבד נראית כך:

```
#include <HardwareSerial.h>
```

```
HardwareSerial SerialPort(2); // use UART2
int LED = 15;
void setup()
{
  SerialPort.begin(115200, SERIAL_8N1, 16, 17);
  pinMode(LED, OUTPUT);
}
void loop()
{
  if (SerialPort.available())
  {
    char number = SerialPort.read();
    if (number == '0') {
      digitalWrite(LED, LOW);
    }
    if (number == '1') {
      digitalWrite(LED, HIGH);
    }
  }
}
```

ט.2.ג איך עובדת התוכנית בעבד ?

הפקודות הראשונות דומות לאלו שבתוכנית המסטר.
* ראשית נכלול את הספרייה הטורית של החומרה :

```
#include <HardwareSerial.h>
```

* יוצרים אובייקט טורי של ספריית HardwareSerial בשם "SerialPort". אנחנו מציינים את מספר פורט ה UART כפרמטר בתוכו. נשתמש ב- UART2 במקרה זה : HardwareSerial SerialPort(2);
* בתוך הפונקציה ה setup() נפתח את התקשורת הטורית של יציאה UART2 בעזרת הפקודה :

```
SerialPort.begin (BaudRate, SerialMode, RX_pin, TX_pin).
```

אנחנו עובדים בקצב תקשורת 115200 ביטים בשנייה, 8 ביטים ללא ביט זוגיות ועם ביט סיום 1. ההדק 16 הוא קליטה ו 17 לשידור.

```
SerialPort.begin(115200, SERIAL_8N1, 16, 17);
```

* נגדיר משתנה מטיפוס שלם ששמו LED ונאתחל אותו בערך 15 שזהו מספר ההדק ב ESP32 אליו מחוברת הלד.

```
int LED = 15;
```


* נקבע את הדק 15 שבו מחוברת הלד כהדק פלט בעזרת הפונקציה pinMode :

```
pinMode(LED, OUTPUT);
```

* בפונקציית () setup , כמו בזו של המסטר , נפתח את התקשורת הטורית של יציאה UART2 באמצעות הפקודה :

```
SerialPort.begin(115200, SERIAL_8N1, 16, 17);
```

כדאי לשים לב שקצב התקשורת גם במסטר וגם בעבד שווים : 115200 ביטים לשנייה.

* כמו כן, נגדיר את הדק ה-LED כהדק פלט באמצעות הפונקציה () pinMode .

```
pinMode(LED, OUTPUT);
```

* בתוך פונקציית ה () loop נבדוק אם יש קליטה טורית על ידי בדיקה האם קיימים נתונים בחוצץ (BUFFER) הקליטה :

```
if (SerialPort.available())
```

* אם כן נקלוט את התו למשתנה תווי בשם number :

```
char number = SerialPort.read();
```

* אם הערך שנקלט ל number הוא '0' (0 אסקי = 0x30) - נכבה את נורית ה-LED

* אם הערך הוא '1' (ערך אסקי 0x31) - נורית ה-LED תופעל.

* היות והמסטר שולח '1' ו-'0' בהשעייה של 5 שניות לעבד. לפיכך, נורית ה-LED נדלקת למשך 5 שניות ולאחר מכן תכבה למשך 5 שניות.

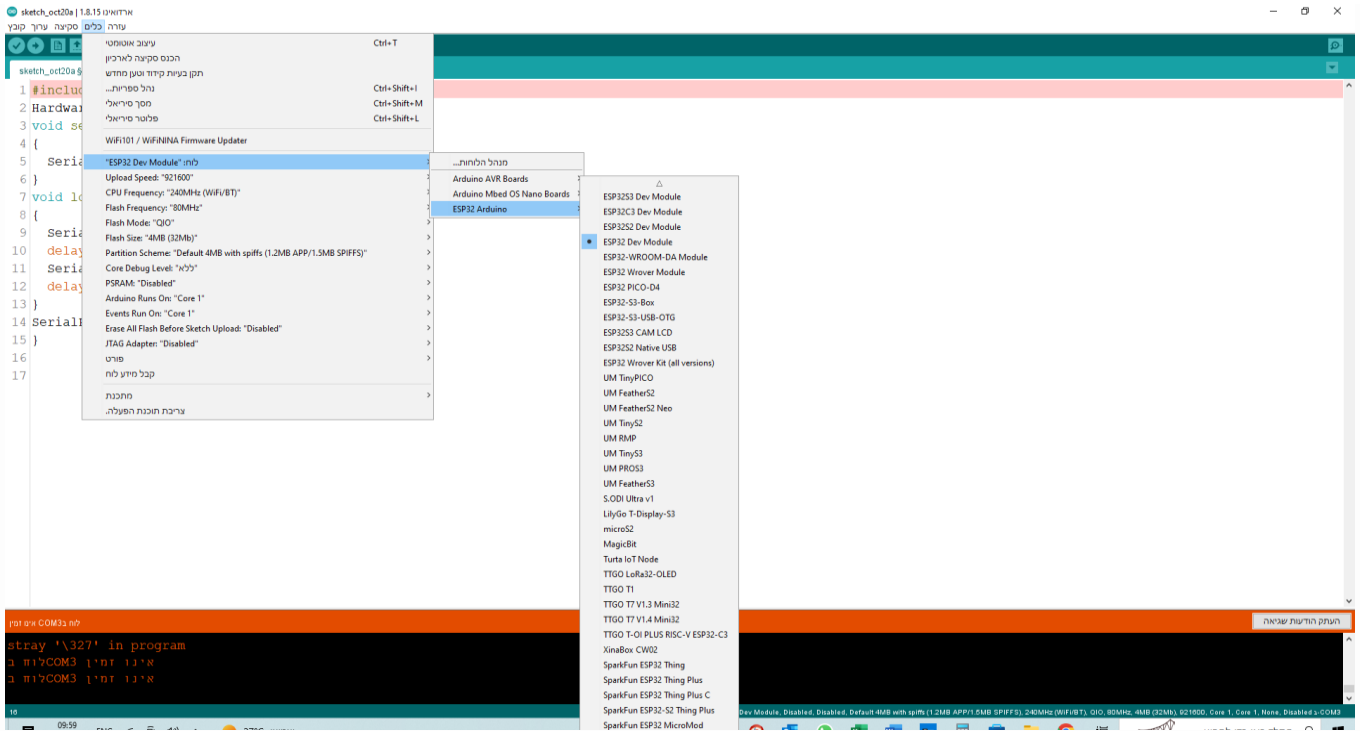
י. הפעלה ובדיקה

כדי לראות שהתוכניות עובדות יש לטעון (upload) את התוכניות של המסטר והעבד לשני לוחות ESP32.

לפני העלאת התוכניות צריך לבחור את מודול ESP32 מתוך Board > Tools (כלים) <- לוח) וגם לבחור את יציאת ה-

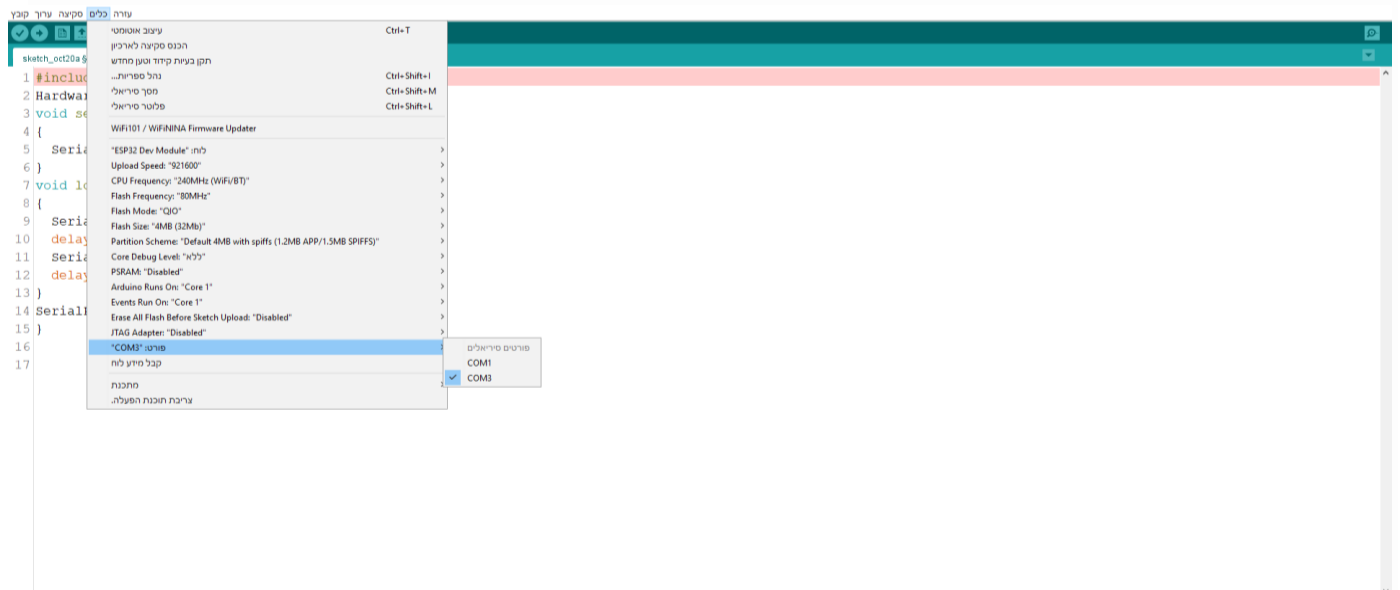
COM הנכונה שאליה מחובר הלוח מ- Port > Tools (כלים) <- פורט).

נעזר באיור הבא כדי לבחור את כרטיס ה-ESP32 המתאים שלך :



איור 4 : בחירת כרטיס ה ESP32 המתאים

כדי לבחור אתה COM המתאים נעזר באיור הבא :



איור 5 : בחירת ה COM המתאים .

לאחר טעינת התוכניות המתאימות גם במסטר וגם בעבד רואים שהלד בכרטיס העבד מהבהבת : 5 שניות דולקת ו 5 שניות נכבית.

יא. דוגמה לתוכנית נוספת המשתמשת ב UART2

```
#define RXD2 16
#define TXD2 17
void setup()
{
  Serial.begin(115200); // אתחול התקשורת עם המוניטור הטורי
  // אתחול תקשורת טורית מספר 2 לקצב 9600 ביטים בשנייה, 8 ביטים, ללא זוגיות
  // עם ביט סיום אחד ומספרי הדק הקליטה והשידור בהתאמה
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
  Serial.println("Serial Txd is on pin: "+String(TX));
  Serial.println("Serial Rxd is on pin: "+String(RX));
}
void loop()
{
  בחר בין Serial או Serial2 לפי הנדרש
  while (Serial2.available())
  {
    Serial.print(char(Serial2.read()));
  }
}
```

התוכנית הזו קוראת נתונים מ- serial2 ושולחת להדפסה ב serial0 שהוא המוניטור/מסך הטורי של הארדואינו. דוגמה לזה יכולה להיות מערכת הקולטת נתונים ומדפיסה אותם למוניטור הטורי של הארדואינו.

ניתן גם להפוך בין התקשורת ולרשום :

```
while (Serial.available())
{
  Serial2.print(char(Serial.read()));
}
```

ואז לשלוח עם המוניטור הטורי של הארדואינו טקסט ולשדר את מה שקלטנו מהמוניטור בעזרת הרכיב הנמצא ב UART2.

יב. HardwareSerialh - הקצאת הדקים שונים מברירת המחדל

בעת עבודה עם Arduino IDE, ניגשים ליציאות הטוריות דרך המחלקה Serial (Serial, Serial1, Serial2). עם זאת, מעבד ESP32 מאפשר למפות את שלוש היציאות הטוריות (UART) לכל פין בין GPIO0 ל-GPIO31. רוב לוחות הפיתוח של ESP32 (כולל מודול אפיק ESP32 CAN) מציעים יציאות נוספות המסומנות TX2/RX2 או דומה, אך אין צורך להשתמש בהדקים אלה במדויק. כל פין GPIO אחר יכול לשמש כ-Serial RX/TX. ל-ESP32 יש Cross Bar (בדומה לרכיבים במשפחת Silicon Labs כמו C8051F380) וניתן להקצות הדקים אחרים עבור התקשורת הטורית ולא חייבים את ההדקים של ברירת המחדל! הדקים אלה מנוהלים באמצעות המחלקה HardwareSerial. הבנאי Constructor שלה מקבל פרמטר אחד שהוא מספר ה-UART. באתחול האובייקט משתמשים בארבעה פרמטרים: קצב שידור טורי, מצב UART, הקצאת הדק ל-RX והקצאת הדק ל-TX.

יב.1 איך משנים הדקי UART ?

לדוגמה שינוי הדקים של UART1 נשנה את הדקי ברירת המחדל :

```
Serial1.begin(9600, SERIAL_8N1, 33, 32);
```

אתחול תקשורת טורית UART1 לקצב 9600 ביטים בשנייה, פרוטוקול התקשורת הוא של 8 ביטים, ללא ביט זוגיות עם ביט סיום 1. הדק RX הוא 33 והדק TX הוא 32.

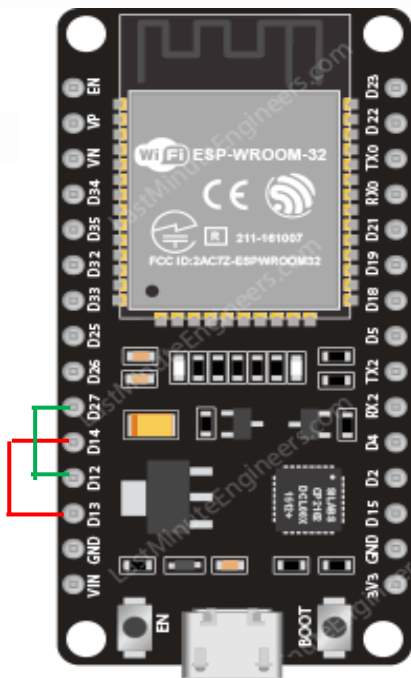
במיקרו פייתון נרשום :

```
uart1 = UART (1, baudrate=9600, tx=33, rx=32)
```

מכאן והלאה, כל שאר פונקציות הטורי (כגון קריאה, כתיבה) תואמות לחומרת Arduino סטנדרטית.

יב.2 דוגמה : תוכנית המשדרת לעצמה וקולטת נתונים מעצמה.

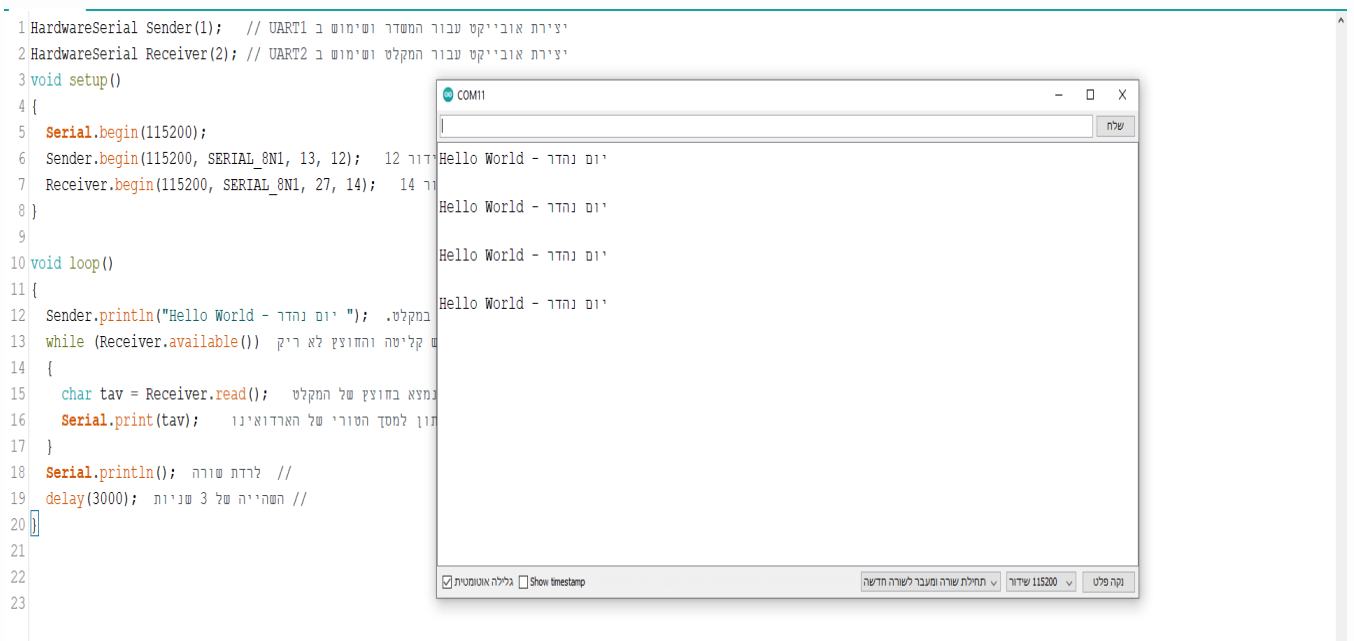
לשם כך נשתמש בחומרה שבאיור הבא:



איור 6 : שידור וקליטה באותו כרטיס

```
HardwareSerial Sender(1); // UART1 עבור המשדר ושימוש ב UART1
HardwareSerial Receiver(2); // UART2 עבור המקלט ושימוש ב UART2
void setup()
{
  Serial.begin(115200);
  Sender.begin(115200, SERIAL_8N1, 13, 12); // 12 ורגל השידור 13
  Receiver.begin(115200, SERIAL_8N1, 27, 14); // 14 ורגל השידור 27
}
void loop()
{
  Sender.println("Hello World - יום נהדר"); // 27 במקלט
  while (Receiver.available()) // נכנסים ללולאה כל עוד יש קליטה והחוצץ לא ריק
  {
    char tav = Receiver.read(); // קרא את הנתון שנמצא בחוצץ של המקלט
    Serial.print(tav); // הדפס את הנתון למסך הטורי של הארדואינו
  }
  Serial.println(); // לרדת שורה
  delay(3000); // השהייה של 3 שניות
}
```

התוצאה המתקבלת בהרצת התוכנית:



איור 7 : מסך המוניטור המתקבל בהרצת התוכנית

1. <https://microcontrollerslab.com/esp32-uart-communication-pins-example/>
2. <https://circuits4you.com/2018/12/31/esp32-hardware-serial2-example/>
3. <https://www.youtube.com/watch?v=eUPAoP7xC7A>