

ESP32 ובלוטות עם ארדואינו IDE

א. מבוא

הנאמר במדריך זה נעזר בקישורים הבאים :

[ESP32 Bluetooth Classic with Arduino IDE - Getting Started | Random Nerd Tutorials](https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/bluetooth.html)

<https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/bluetooth.html>

https://deepbluembedded.com/esp32-bluetooth-classic-with-arduino-complete-guide/#google_vignette

<https://www.mathworks.com/help/bluetooth/gs/comparison-of-bluetooth-bredr-and-bluetooth-le.html>

<https://www.bluetooth.com/>



1.א מעט על בלוטות

על בלוטות ועל מודולים של בלוטות כמו HC05 או HC06 וחיבורם אל ארדואינו ניתן לקרוא בקישור :

<https://www.arikporat.com/wp-content/uploads/2022/12/hc-05-bluetooth.pdf>

נסביר כאן בקצרה על הטכנולוגיה ועל ההקשר אל ESP32 .

טכנולוגיית Bluetooth - בלוטות - פועלת בתחום התדרים התעשייתי, המדעי והרפואי (ISM) ללא רישיון בתדר של 2.4 GHz, ומשתמשת בתדרי רדיו בהספק נמוך כדי לאפשר תקשורת לטווח קצר בעלות נמוכה. שתי הגרסאות של טכנולוגיית Bluetooth הן :

- Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) or classic Bluetooth

שפרושו בלוטות קצב בסיסי/ מועשר או בלוטות קלאסי .

- Bluetooth Low Energy (BLE) or Bluetooth Smart

שפרושו בלוטות אנרגיה נמוכה (BLE) או בלוטות חכם .

מפרט הליבה של Bluetooth שצוין על-ידי קונסורציום/איגוד קבוצת העניין המיוחדת Special Interest Group (SIG) , מגדיר את הטכנולוגיות הדרושות ליצירת התקני Bluetooth BR/EDR ו- Bluetooth LE בעלי יכולת פעולה הדדית.

Bluetooth BR/EDR מיועד בעיקר לפעולות בהספק נמוך ובתפוקת נתונים גבוהה. ב- Bluetooth BR / EDR, תדר ה RF קופץ בצורה פסאודו-אקראית על 79 ערוצי Bluetooth ייעודיים. לכל ערוץ Bluetooth BR/EDR יש רוחב פס של 1 MHz. כל תדר ממוקם ב $(2402 + k)$ MHz, כאשר $k = 0, 1, \dots, 78$. כלומר מתחיל ב 2402 MHz (2.402GHz) ואחרי 2.403GHz עד 2.48GHz .

בשנת 2010, SIG הציגה את Bluetooth LE (בלוטות עם אנרגיה נמוכה) עם גרסת Bluetooth 4.0. תדר ה RF ב Bluetooth LE מתוכנן לתמיכה ביישומים בעלי מחזור עבודה נמוך יחסית. לדוגמה, נניח שאדם לובש מכשיר ניטור דופק במשך מספר שעות. מכיון שמכשיר זה משדר רק כמה בתים של נתונים בכל שנייה, השידור שלו נמצא במצב 'מופעל' לפרק זמן קצר מאוד. ב- Bluetooth LE, תדר רדיו ההפעלה הוא בטווח שבין GHz 2.4000 ל- GHz 2.4835. רוחב הפס של הערוץ הוא 2 MHz, ורצועת ההפעלה מחולקת ל-40 ערוצים, $(k = 0, 1, \dots, 39)$. התדר המרכזי של ערוץ kth ממוקם ב $(2402 + k \times 2)$ MHz .

א.2 ESP32 ובלוטות

המיקרו בקר ESP32 מכיל בתוכו גם בלוטות אנרגיה נמוכה (BLE) Bluetooth Low Energy, גם בלוטות קלאסי Bluetooth Classic וגם Wi-Fi. החומרה של ה-ESP32 תומכת ב- BLE v4.2, מה שאומר שהיא אינה תומכת ב- Bluetooth 5.0 כרגע. ניתן להשתמש הן ב-ESP32 BLE והן ב- Bluetooth Classic עבור יישומי קישוריות. הנאמר כאן הוא מדריך מקיף עבור ESP32 Bluetooth Classic. נלמד כיצד להשתמש ב-ESP32 Bluetooth Classic עם Arduino IDE, וכיצד לבצע את כל הפעולות העיקריות כמו (צימוד Bluetooth, סורק Bluetooth, שליחת נתונים במצב מאסטר - Master - וקבלת נתונים במצב עבד - Slave).

ב. בלוטות קלאסי - Bluetooth Classic

זוהי האפשרות הקלה יותר עבור יישומי תקשורת Bluetooth ESP32. זה עובד בדיוק כמו כל מודולי Bluetooth טוריים (UART) שהשתמשנו בהם עם Arduino (כמו HC-05, HC-06 וכו'). Bluetooth Classic משתמש בתחום התדרים של ISM 2.4 GHz (תעשייתית, מדעית ורפואית - ISM – industrial, Scientific and Medical) יש לו קצב נתונים מרבי של 3 Mbps. יש לו טווח של עד 100 מטר בשטח פתוח ועד 10 מטר בתוך הבית. התקני Bluetooth קלאסיים יכולים להתחבר להתקנים אחרים באמצעות מגוון פרוטוקולים כגון - A2DP (פרופיל הפצת שמע מתקדם - Advanced Audio Distribution Profile).

ג. בלוטות אנרגיה נמוכה - ESP32 BLE (Bluetooth Low Energy)

Bluetooth Low Energy (BLE) היא גרסה חדשה יותר של טכנולוגיית Bluetooth שהוצגה בשנת 2010. BLE משתמש באותה רצועת ISM של 2.4 GHz כמו Bluetooth קלאסי, אך הוא צורך פחות חשמל יש לו טווח מופחת. הוא משמש בעיקר עבור יישומים בעלי צריכת חשמל נמוכה שבהם רוצים להעיר את המכשיר מעת לעת, לקרוא נתוני חיישנים, לשלוח אותם באמצעות BLE ולחזור לשינה קלה. השימוש ב-ESP32 BLE הוא קצת יותר מסובך מאשר Bluetooth קלאסי ואנו נסביר אותו במאמר אחר.

ד. השוואה בין בלוטות קלאסי לבלוטות אנרגיה נמוכה

בטבלה הבאה נעשה השוואה מסוכמת מהירה בין ESP32 Bluetooth Classic ו- BLE :

Feature	התכונה	Bluetooth Classic	Bluetooth Low Energy (BLE)
	שימוש	זרימת נתונים ממושכת	לפרץ קצר של נתונים (נתונים לזמן קצר)
	קצב נתונים	1 Mbps for BR** 2-3 Mbps for EDR**	500kbps-1Mbps
	רוחב פס	2.4 GHz ISM band (2400-2483.5	2.4 GHz ISM band (2400-2483.5 MHz)
	מספר ערוצים	79 ערוצים כל אחד ברוחב z	40 ערוצים כל אחד ברוחב 2MHz
	טווח התקשרות	The Same (8m up to 100m)	The Same (8m up to 100m)
	תצרוכת הספק	High (up to 1W)	Low (0.01W up to 0.5W)
	האם הכרחי צימוד/תיאום	YES	NOT Mandatory
	טופולוגיות נתמכות	Peer-to-peer (1:1)	Peer-to-peer (1:1), Star topology Broadcast (1:many), and Mesh (many:m
	טכניקת אפנון	GFSK for BR 8-DPSK or $\pi/4$ -DQPSK for EDR	GFSK
	Latency	35ms	2-16ms (Avg. 9ms)

טבלה 1 : השוואה בין בלוטות קלאסי לבלוטות אנרגיה נמוכה

**

הבלוטות הבסיסי הנקרא BR קיצור של Base Rate תומך בשיטת אפנון הפועלת על שינוי תדר הגל הנושא הנקראת GFSK, כדי להעביר את המידע בקצב העברה של 1 מגה ביט לשנייה.

אחרי הצגת בלוטות' EDR 2.0 + החל בלוטות לתמוך גם בגרסאות של אפנון פאזה הנקראות $\pi/4$ -DQPSK ו-8 DPSK בין התקנים תומכים. המושג EDR קיצור של Enhanced Data Rate (קצב העברה משופר) משמש לתיאור המודולציות $\pi/4$ -DQPSK ו-8 DPSK שתומכות קצב העברה של 2 ו-3 מגה ביט לשנייה בהתאם. השילוב של שני הסוגים הללו יחד נקרא בקיצור BR/EDR.

ה. ממשקי API של הספרייה הטורית

הערה: API זה ראשי תיבות של **Application Programming Interface** - ממשק תכנות יישומים. במונח זה משתמשים בעולם התכנות והטכנולוגיה על מנת לתאר דרך או גישה למידע של שירות חיצוני, בדרך כלל באמצעות קוד. ספריית Bluetooth ESP32 שעלינו לכלול היא "BluetoothSerial.h". זה כולל יישום של פונקציות שימושיות רבות ליצירת פרויקטים עם ESP32 Bluetooth. בסעיף זה, נדון בפונקציות (API) הנפוצות ביותר בספריית Bluetooth ESP32.

ה.1 הכללת הספרייה ואתחול

בתחילת התוכנית נכלול את הספרייה, נבדוק אם Bluetooth מופעל כראוי וניצור אובייקט של BluetoothSerial.

```

1 #include "BluetoothSerial.h"
2
3 #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
4 #error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
5 #endif
6 #if !defined(CONFIG_BT_SPP_ENABLED)
7 #error Serial Bluetooth not available or not enabled. It is only available for the ESP32
8 #endif
9
10 // Bluetooth Serial Object (Handle)
11 BluetoothSerial SerialBT;
```

ה.2 פונקציית ה `setup()`

בפונקציית ה `setup()` נרשום את המשפטים הבאים: כדי להפעיל את מודול Bluetooth נשתמש בפונקציה/מתודה `begin()` ולאחר מכן נוכל לקבוע, אם רוצים, את קוד ה PIN. אפשר לתת להתקן Bluetooth כל שם מחרוזת שנרצה ו/או להקצות קוד PIN להתאמה עם התקנים אחרים.

```

1 void setup() {
2 ..
3 SerialBT.begin(device_name); // Bluetooth device name
4 SerialBT.setPin(pin); // If you want to use a PIN
5 ..
6 }
```

כמובן שכדאי לאתחל את התקשורת הטורית עם המוניטור/מסך טורי תוך פעולות אתחול נדרשות (לא הראינו בקטע קוד זה).

ה.3 שליחה וקבלת נתונים

כדי לשלוח נתונים באמצעות Bluetooth, נשתמש בפונקציה/מתודה write () בדיוק כמו serial.write() עבור תקשורת טורית.

```
1 SerialBT.write(TxBuffer); // Send Data Over Bluetooth
```

כדי לקרוא נתונים מחוצץ (Buffer) שהתקבל על-ידי Bluetooth נבדוק אם יש נתונים זמינים בחוצץ, ואם כן נקרא את הנתונים באמצעות הפונקציה/מתודה read(). נמשיך ונקרא את הנתונים מהחוצץ של התקשורת הטורית, בית אחרי בית, אל מחרוזת שנגדיר בשורה הראשונה בתוכנית שבהמשך (String RxBuffer = ""); עד לקבלת התו '\n' שמציין שורה חדשה.

```
1 String RxBuffer = "";
2 ..
3 if (SerialBT.available()){
4   RxByte = SerialBT.read();
5   if (RxByte != '\n'){
6     RxBuffer += String(RxByte);
7   }
8   else{
9     RxBuffer = "";
10  }
11 }
```

ה.4 קבלת כתובת ה MAC של הבלוטות

כתובת MAC באנגלית - **Media Access Control address** - כתובת בקרת גישה למדיה - היא מזהה ייחודי המוטבע על כל רכיב תקשורת נתונים בעת הייצור.

כדי לקבל את כתובת התקן Bluetooth ESP32, עלינו להשתמש בפונקציה/מתודה getBtAddressString(). פונקציה זו אינה מקבלת ארגומנטים ומחזירה את כתובת Bluetooth של שישה בתים כמחרוזת שבאפשרותנו להדפיס לצג הטורי. הנה דוגמה כיצד לעשות זאת ב-Arduino IDE.

```
1 #include "BluetoothSerial.h"
2
3 BluetoothSerial SerialBT;
4 String MAC_Address; // MAC הגדרת מחרוזת אליה תיכנס כתובת ה
5
6 void setup() {
7   Serial.begin(115200);
8   SerialBT.begin("ESP32 Bluetooth");
```

```

9 }
10
11 void loop() {
12   MAC_Address = SerialBT.getBtAddressString();
13   Serial.println(MAC_Address.c_str());
14   delay(500);
15 }

```

ה.5 המתודות להתחברות לבלוטות Master.connect() ולהתנתקות Master.disconnect

במצב Bluetooth Master יש להתחבר להתקן Bluetooth עבד (Slave) לפני שניתן יהיה להחליף נתונים. לשם כך, נזדקק לפונקציה/מתודה connect() הכוללת שתי גרסאות כדי לתמוך בהתחברות להתקן עבד ידוע בשמו או בכתובת MAC שלו. פעולה זו תחזיר את מצב החיבור בין אם הוא הצליח או נכשל.

```

1 connected = SerialBT.connect(SlaveName); // Use Name
2 connected = SerialBT.connect(SlaveAddress); // Or MAC Address

```

כדי לסיים את התקשורת עם התקן העבד, נשתמש בפונקציה disconnect(). סיום ההתקשרות יכול להימשך עד 10 שניות.

```

1 // Disconnect() may take up to 10 secs max
2 if (SerialBT.disconnect()) {
3   // Disconnected Successfully!
4 }

```

ה.6 הפונקציה ESP32 Bluetooth Events Callback

במקום לבצע שאילתה (POLLING) על אירועי Bluetooth שונים ולהשאיר את המעבד חסום/עסוק בהמתנה לקבלת נתונים מסוימים, או לסגור כל אירוע Bluetooth אחר, אנו יכולים במקום זאת להשתמש בפונקציית Callback כדי לקבל התראה כאשר אירוע Bluetooth כלשהו מתרחש.

בעיקרון, זוהי פונקציה שנגדיר ותגרום למנהל ההתקן של BluetoothSerial להצביע עליה. כאשר מתרחש אירוע Bluetooth כלשהו, פונקציית ההתקשרות החוזרת תיקרא ותבצע. לכן, באחריותנו לבדוק איזה אירוע התרחש ולטפל בו בהתאם. להלן האירועים הזמינים שנוכל לבדוק בפונקציית ההתקשרות החוזרת (כל אירוע מהם יפעיל התקשרות חוזרת).

ESP_SPP_INIT_EVT: When SPP mode is initialized (כאשר מצב SPP מאותחל)

ESP_SPP_UNINIT_EVT: When the SPP mode is deinitialized (כאשר מצב SPP לא מאותחל)

ESP_SPP_DISCOVERY_COMP_EVT: When service discovery is complete (כאשר הושלם שרות הגילוי)

ESP_SPP_OPEN_EVT: When an SPP client opens a connection (כאשר לקוח SPP פותח חיבור)

ESP_SPP_CLOSE_EVT: When an SPP connection is closed (כאשר חיבור SPP נסגר)

ESP_SPP_START_EVT: When the SPP server is initialized (כאשר שרת SPP מאותחל)

ESP_SPP_CL_INIT_EVT: When an SPP client initializes a connection (כשלקוח SPP מאתחל חיבור)

ESP_SPP_DATA_IND_EVT: When receiving data through an SPP connection (כשמקבלים נתונים באמצעות חיבור SPP)

ESP_SPP_CONG_EVT: When congestion status changes on an SPP connection (SPP משתנה בחיבור)

ESP_SPP_WRITE_EVT: When sending data through SPP. (בעת שליחת נתונים בעזרת SPP)

ESP_SPP_SRV_OPEN_EVT: When a client connects to the SPP server (כאשר לקוח מתחבר לשרת SPP)

ESP_SPP_SRV_STOP_EVT: When the SPP server stops (כאשר שרת SPP נעצר)

על מנת להשתמש במתודות אלו במקום לבצע תשאול (POLLING) לאירועים, עלינו לבצע 2 שלבים. קודם כל, להגדיר פונקצייה המטפלת באירועים Callback עבור Bluetooth בהתאם ליישום שלנו. אין צורך לבדוק ולטפל בכל האירועים, אלא רק באלה שאנחנו מעוניינים בהם. השלב השני הוא לגרום למנהל ההתקן (דרייבר) של BluetoothSerial להצביע על הפונקציה Callback שהגדרנו בשלב הראשון. נראה דוגמה איך לעשות את זה.

```

1 ...
2
3 // Bluetooth Event Handler Callback Function Definition
4 void BT_EventHandler(esp_spp_cb_event_t event, esp_spp_cb_param_t *param) {
5     if (event == ESP_SPP_START_EVT) {
6         Serial.println("Initialized SPP");
7     }
8     else if (event == ESP_SPP_SRV_OPEN_EVT ) {
9         Serial.println("Client connected");
10    }
11    else if (event == ESP_SPP_CLOSE_EVT ) {
12        Serial.println("Client disconnected");
13    }
14    else if (event == ESP_SPP_DATA_IND_EVT ) {
15        Serial.println("Data received");
16        while (SerialBT.available()) {
17            int incoming = SerialBT.read();
18            Serial.println(incoming);
19        }
20    }
21 }
22
23 void setup() {
24     ...
25     SerialBT.begin(device_name);
26     SerialBT.setPin(pin);

```

```
27 // Attach The Callback Function Definition To SerialBluetooth Events
28 SerialBT.register_callback(BT_EventHandler);
29 ...
}
```

1. יישומים עבור ESP32 Bluetooth

בחלק זה נבחן פעולות שונות עבור **ESP32 Bluetooth Classic** וכיצד לבצע אותן ב-Arduino IDE. נראה דוגמה עבור כל יישום שניתן לשנות לפי הצורך לפני הבדיקה.

אנחנו נשלוט בפלט ה-ESP32 ונשלח קריאות חיישנים לסמארטפון אנדרואיד באמצעות Bluetooth Classic. **הערה:** פרויקט זה תואם רק לסמארטפונים של Android. ניתן לראות סרטון הדרכה בקישור:

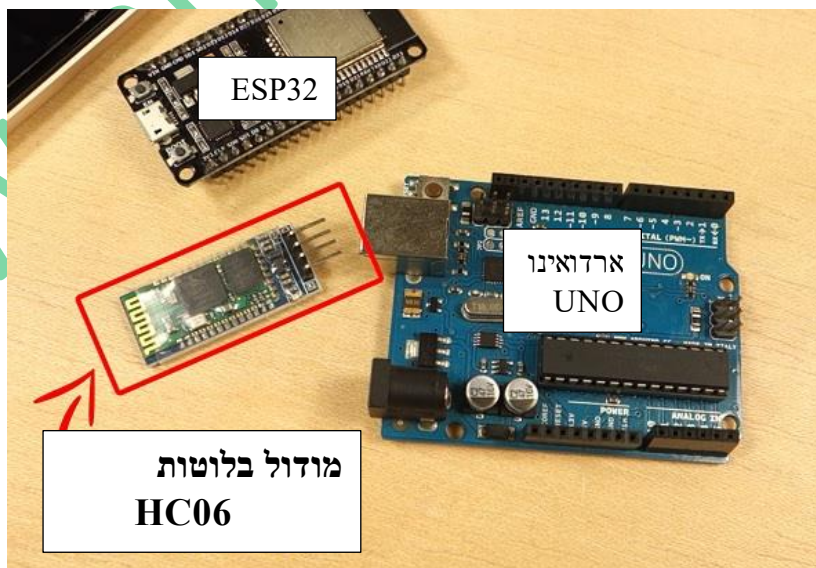
<https://youtu.be/RStncO3zb8g>

2. בלוטות קלאסי עם ESP32

כרגע, השימוש ב-Bluetooth Classic הוא הרבה יותר פשוט מאשר Bluetooth Low Energy. אם כבר התנסו בחיבור Arduino עם מודול Bluetooth כמו HC-06, זה דומה מאוד. הוא משתמש בפרוטוקול הטורי הסטנדרטי ובפונקציות. האיור הבא מתאר כרטיס מיקרו בקר של ארדואינו אונו וכרטיס ESP32 וגם מודול בלוטות HC06. הסבר מפורט על מודול בלוטות כמו HC05 ו HC06 וכיצד מצמידים 2 מודולים של בלוטות נמצא בקישורים:

<https://www.arikporat.com/wp-content/uploads/2022/12/hc-05-bluetooth.pdf>

<https://www.arikporat.com/wp-content/uploads/2022/12/hc05-pairing.pdf>



איור 1: מיקרו בקרים ארדואינו, ESP32 ומודול בלוטות HC06.

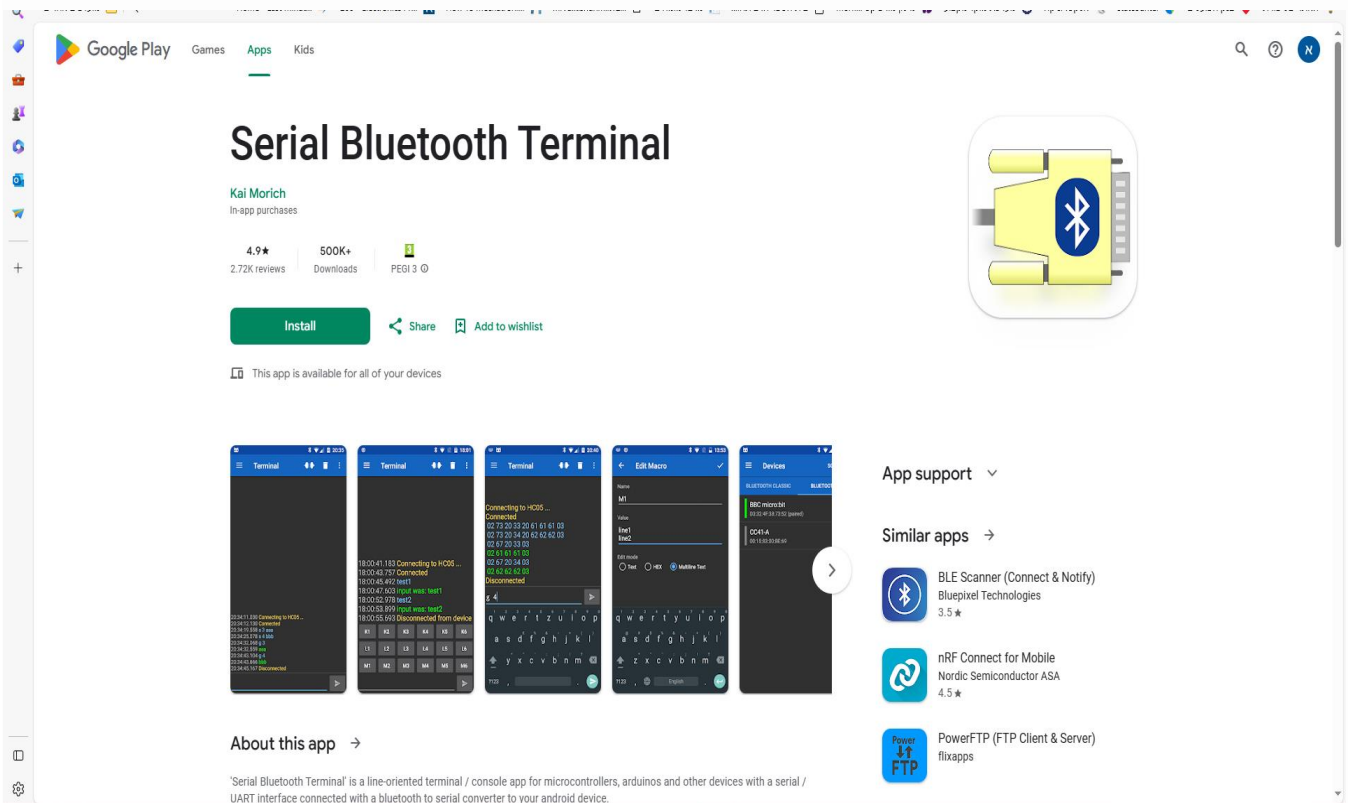
נתחיל בדוגמה פשוטה שמגיעה עם סביבת העבודה של Arduino IDE שבה משתמשים במיקרו בקר ESP32 בתקשורת בלוטות אל הטלפון הנייד שלנו, כאשר בטלפון נתקין את האפליקציה Bluetooth Terminal . אין צורך במודולים/רכיבים חיצוניים.

ת. האפליקציה Bluetooth Terminal

תחילה יש להתקין יישום Bluetooth Terminal בטלפון הנייד . מומלץ להשתמש באפליקציית Android

[Serial Bluetooth Terminal](#)

הזמין בחנות google Play ונראה באיור הבא :



איור 2 : האפליקציה Serial Bluetooth Terminal
יש ללחוץ על Install ולהתקין את האפליקציה בטלפון הנייד.

ט. בלוטות טורי לטורי

אנחנו נתכנת את המיקרו ESP32 בסביבת העבודה של Arduino IDE . יש לוודא שהתקנו את התוספת ESP32 לפני שנמשיך. ניתן למצוא הסבר על חיבור התוספת ESP32 בסביבת ארדואינו בקישור :

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

הסבר באנגלית ניתן למצוא בקישור :

- [Windows: instructions – ESP32 Board in Arduino IDE](#)

ואתו ההסבר במערכות MAC ו LINUX נמצא בקישור :

- [Mac and Linux: instructions – ESP32 Board in Arduino IDE](#)

נפתח את תוכנת הארדואינו ונעבור אל :

File > Examples > BluetoothSerial > SerialtoSerialBT

נקבל את הקוד הבא :

```
//This example code is in the Public Domain (or CC0 licensed, at your option.)
//By Evandro Copercini - 2018
//
//This example creates a bridge between Serial and Classical Bluetooth (SPP)
//and also demonstrate that SerialBT have the same functionalities of a normal Serial

#include "BluetoothSerial.h"

#ifdef USE_PIN // Uncomment this to use PIN during pairing. The pin is specified on the line below
const char *pin = "1234"; // Change this to more secure PIN.

String device_name = "ESP32-BT-Arik Porat";

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  SerialBT.begin(device_name); //Bluetooth device name
  Serial.printf("The device with name \"%s\" is started.\nNow you can pair it with Bluetooth!\n",
device_name.c_str());
```

```
//Serial.printf("The device with name \"%s\" and MAC address %s is started.\nNow you can pair it with Bluetooth!\n", device_name.c_str(), SerialBT.getMacString()); // Use this after the MAC method is implemented
```

```
#ifdef USE_PIN
    SerialBT.setPin(pin);
    Serial.println("Using PIN");
#endif
}
```

```
void loop() {
    if (Serial.available()) {
        SerialBT.write(Serial.read());
    }
    if (SerialBT.available()) {
        Serial.write(SerialBT.read());
    }
    delay(20);
}
```

י. הסבר התוכנית

התוכנית יוצרת תקשורת Bluetooth טורית דו-כיוונית בין שני התקנים. בדוגמה כאן בין ESP32 לטלפון הנייד. הקוד מתחיל בהכללת ספריית BluetoothSerial.

```
#include "BluetoothSerial.h"
```

השורה הבאה מסומנת כהערה אבל יש לבטל הערה זו כדי להשתמש בקוד PIN במהלך השיך. ה PIN (קוד הצימוד) רשום כקבוע בשורה הבאה. הוא ברוב המקרים "1234" וכדאי לשנות אותו למאובטח יותר.

```
///define USE_PIN // Uncomment this to use PIN during pairing. The pin is specified on the line below
const char *pin = "1234"; // Change this to more secure PIN.
```

המשפט הבא :

```
String device_name = "ESP32-BT-Arik Porat";
```

מגדיר מחרוזת שתציין את השם של הבלוטות שייראה בצד השני (בטלפון הנייד) בזמן הצימוד/תאום.

שלוש השורות הבאות :

```
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif
```

נקראות הנחיות קדם מעבד - Pre processor – והן בודקות אם Bluetooth מופעל כראוי והבלוטות זמין. בשורות אלו בודקים האם לא הוגדר (CONFIG_BT_ENABLED) או לא הוגדר CONFIG_BLUEDROID_ENABLED , ובמקרה שאחד או שניהם לא הוגדרו נקבל הדפסת שגיאה בזמן הקומפילציה, שגיאה שבה נאמר שהבלוטות לא מאופשר ויש להריץ **menuconfig** כדי לאפשר אותם.

3 השורות הבאות בתוכנית :

```
#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif
```

גם הן הוראות קדם מעבד הבודקות האם הוגדר הבלוטות עבור ה ESP32 . גם כאן נקבל הודעת שגיאה בזמן הקומפילציה , אם לא הייתה הכללה של CONFIG_BT_SPP_ENABLED שאומרת שה Serial Bluetooth אינו זמין או אינו מופעל. הוא זמין רק עבור שבב/רכיב ESP32.

לאחר מכן יוצרים אובייקט של BluetoothSerial בשם SerialBT; BluetoothSerial SerialBT;

1.י בפונקציית ה setup()

```
Serial.begin(115200);
```

מאתחלים את קצב התקשורת עם המוניטור הטורי של הארדואינו ל 115200 ביטים בשנייה.

בשורה הבאה מאתחלים את ההתקן הטורי של Bluetooth ומעבירים לו כארגומנט את שם התקן Bluetooth. אני קראתי לו "ESP32-BT-Arik Porat" אך ניתן לשנות את שמו ולהעניק לו כל שם ייחודי שנרצה.

```
SerialBT.begin(device_name); //Bluetooth device name
```

בשורה הבאה :

```
Serial.printf("The device with name \"%s\" is started.\nNow you can pair it with Bluetooth!\n", device_name.c_str());
```

מדפיסים למסך הטורי מחרוזת שבה מודפס שהרכיב עם השם device_name (כאן "ESP32-BT-Arik Porat") התחיל. ובשורה הבאה מודפס שניתן לצמד אותו עכשיו עם Bluetooth .

השורה הבאה רשומה כהערה .

```
//Serial.printf("The device with name \"%s\" and MAC address %s is started.\nNow you can pair it
with Bluetooth!\n", device_name.c_str(), SerialBT.getMacString()); // Use this after the MAC method
is implemented
```

השורות הבאות :

```
#ifdef USE_PIN
  SerialBT.setPin(pin);
  Serial.println("Using PIN");
#endif
```

הן שוב הנחיות קדם מעבד. הן אומרות שאם הגדרנו USE_PIN (סיסמה)

2. פונקציית ה loop()

נשלח ונקבל נתונים באמצעות Bluetooth Serial. יש לנו 2 משפטי if .

במשפט ה if הראשון בודקים האם כתבנו משהו במוניטור הטורי ואם כן משדרים את מה שכתבנו בתקשורת בלוטות לצד השני. במשפט ה if השני בודקים האם הצד השני שידר אלינו נתון. אם כן אנחנו קוראים אותו מהבלוטות וכותבים אותו במוניטור הטורי.

השליחה מתבצעת על ידי כתיבה במסך הטורי/מוניטור של הארדואינו ולחיצה על Enter או send. כל תו שנרשום במוניטור הטורי ייקלט על ידי התקשורת הטורית של ה ESP32 ויכתב אל רכיב הבלוטות שישדר אותו אל הטלפון הנייד שלנו או לרכיב בלוטות שעשינו איתו צימוד/תיאום - pairing. במשפט ה if הראשון, אנו בודקים אם נקלטו בתים בפורט הטורי של המוניטור. אם כן נשלח מידע זה באמצעות Bluetooth להתקן המחובר.

```
if (Serial.available()) // האם נקלטו דרך הפורט הטורי נתונים ?
{
  SerialBT.write(Serial.read()); // אם כן שלח את מה שקלטנו לבלוטות
}
```

המשפט Serial.available() בודק האם התקבל נתון כלשהו מהמוניטור הטורי של הארדואינו. אם כן נכנסים למשפט שאומר לשלוח את הנתון שהתקבל/שנקרא - Serial.read() ולכתוב אותו אל הבלוטות SerialBT.write .

משפט ה if השני עושה פעולה הפוכה. בודקים האם נקלטו נתונים כלשהם דרך הפורט הטורי של הבלוטות (נתון שהצד השני שידר אלינו). אם כן אנחנו כותבים למוניטור הטורי את הנתון שאנחנו קוראים מהבלוטות.

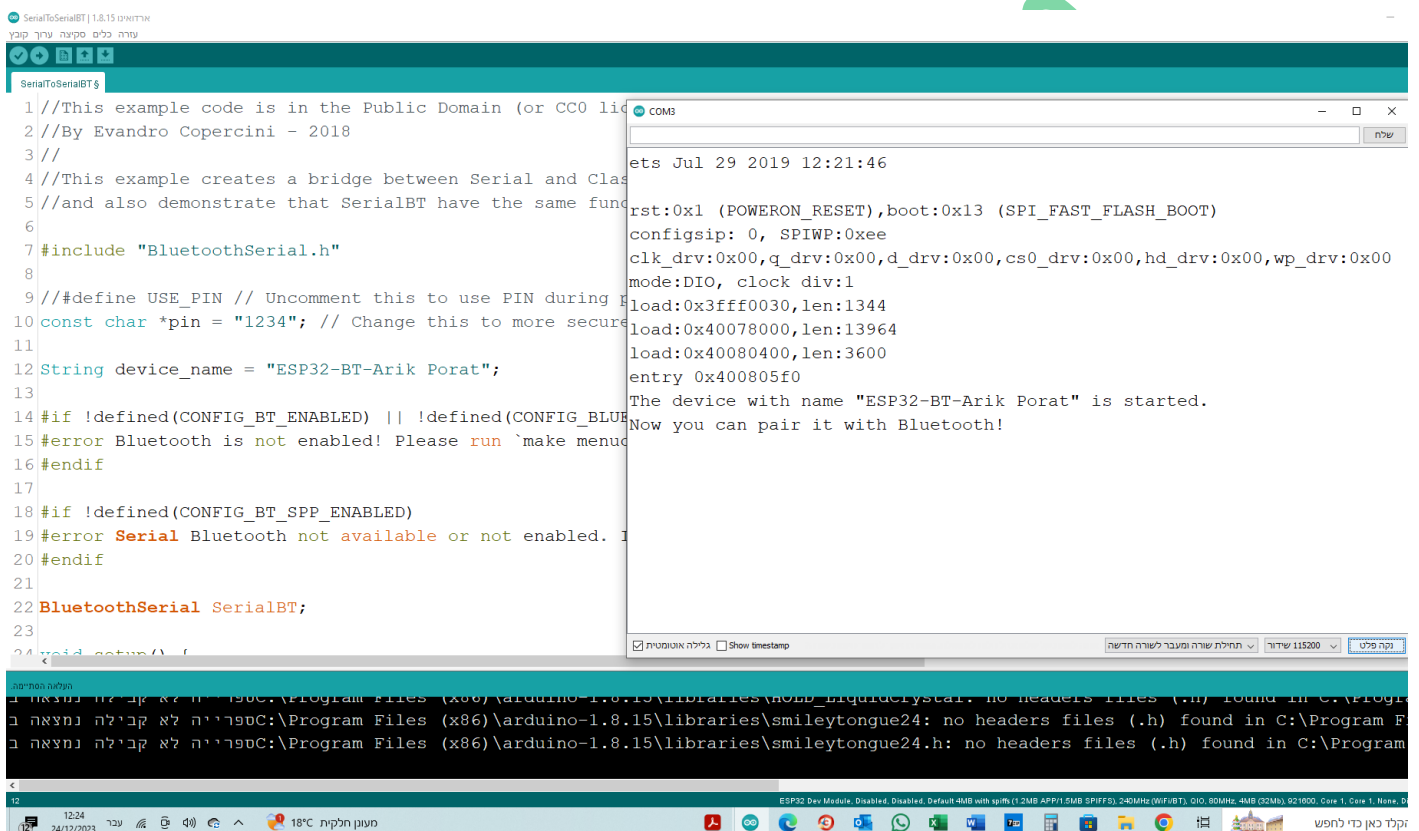
```
if (SerialBT.available()) { // האם נקלט נתון כלשהו בתקשורת בלוטות ?
  Serial.write(SerialBT.read()); // אם כן כתוב למוניטור הטורי את הנתון שקוראים מהבלוטות
}
```

יהיה קל יותר להבין בדיוק איך התוכנית עובדת בעזרת דוגמה.

3. דוגמה

נטען את התוכנית ל-ESP32. נוודא שבחרנו את הלוח הנכון ואת יציאת ה-COM. לאחר העלאת הקוד, נפתח את הצג הטורי בקצב שידור של 115200. נלחץ בכרטיס ה-ESP32 על הלחצן ESP32 Enable (הפעל את ESP32). לאחר מספר שניות, נקבל הודעה שאומרת: *"The device started, now you can pair it with bluetooth!"* כלומר "הרכיב התחיל, עכשיו אתה יכול לתאם אותו עם Bluetooth!".

באיור הבא מתאר את המתקבל:



איור 3 : המסך המתקבל שאומר שהבלוטות הותחל וניתן לבצע צימוד.



Domain (or CC0 lic

```
ets Jul 29 2019 12:21:46
Hello my students
rst:0x1 (POWERON_F
configsip: 0, SPIW
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0
The device with name "ESP32-BT-Arik Porat" is started.
Now you can pair it with Bluetooth!
```

een Serial and Clas
have the same func

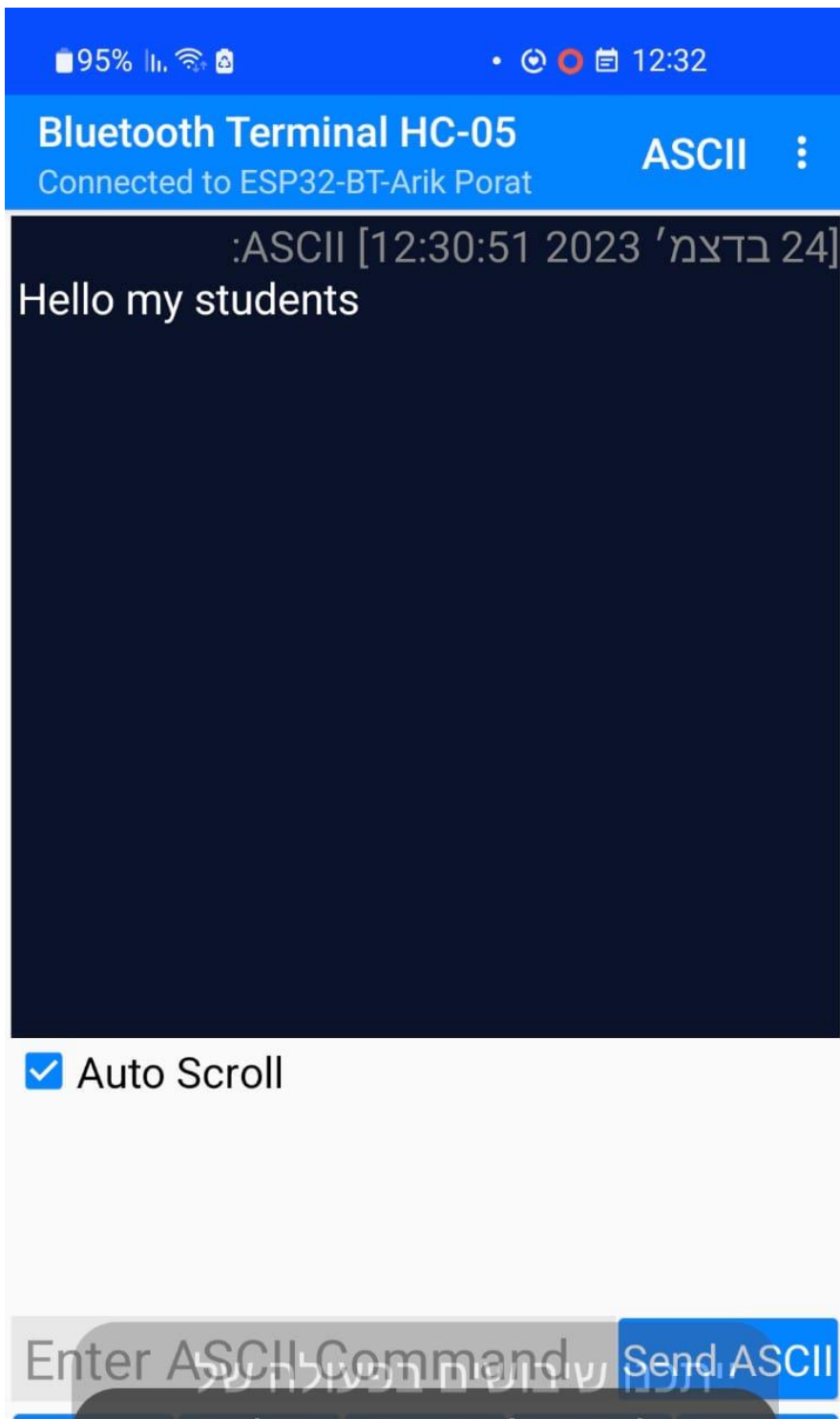
to use PIN during P
this to more secure

Porat";

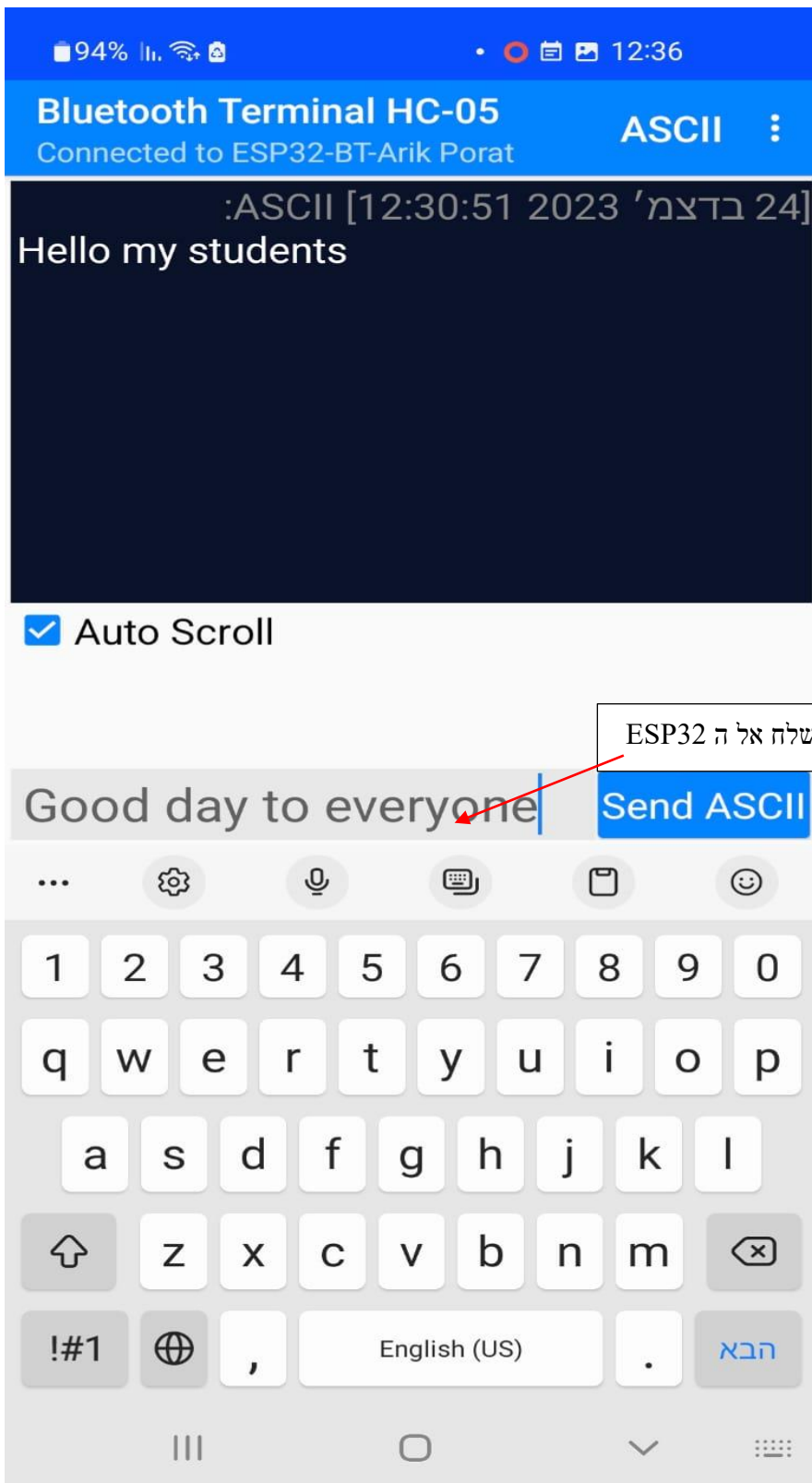
defined(CONFIG_BLUE
ase run `make menu

איור 4 : שליחת הודעה אל הנייד

ונקבל בנייד :

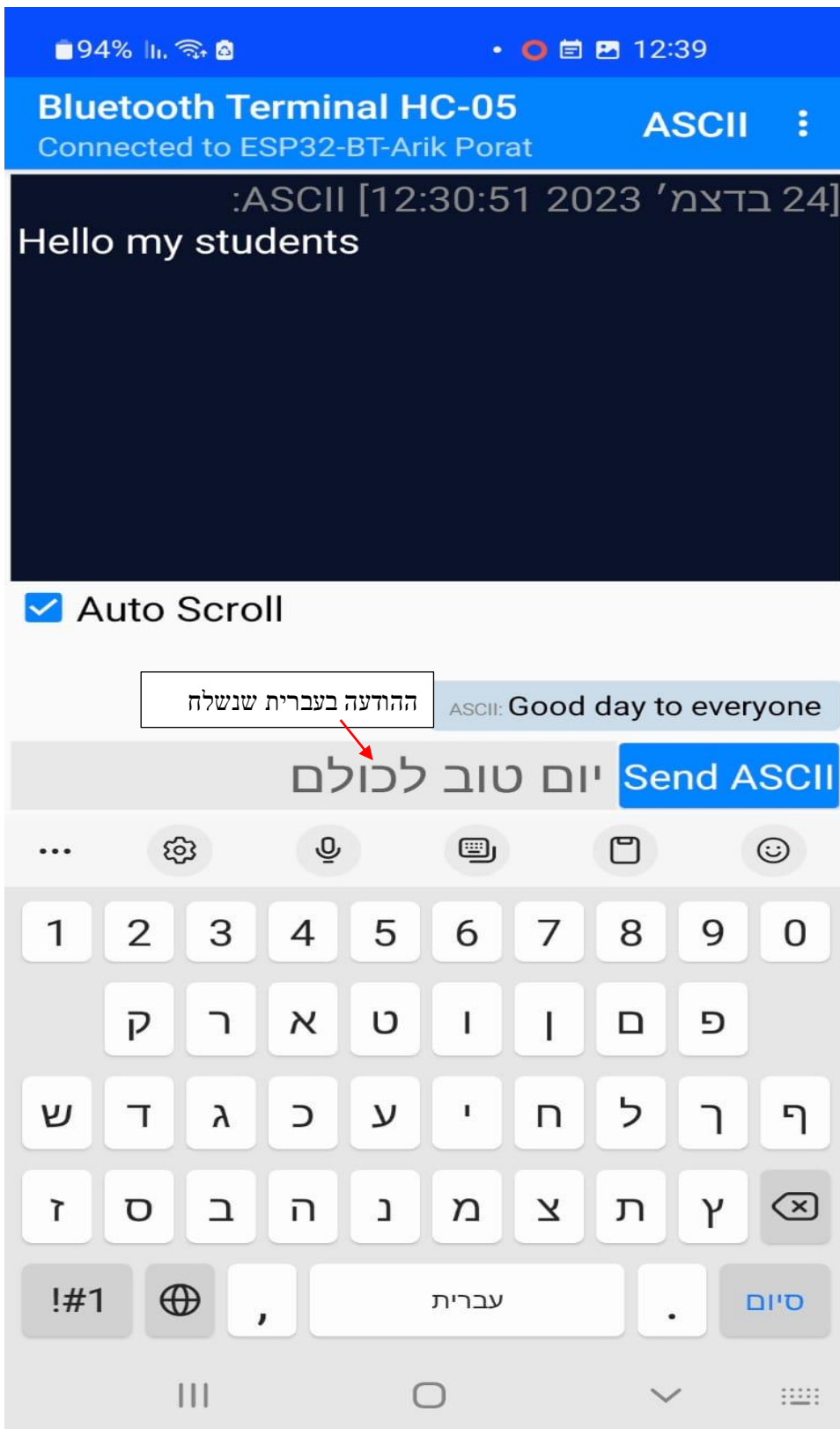


עכשיו נשלח הודעה מהטלפון הנייד אל ה-ESP32. ההודעה שנשלח באנגלית היא: Good day to everyone



ואז נלחץ על Send ASCII וב IDE של הארדואינו, במוניטור הטורי נקבל את ההודעה במסך.

אם נרשום את ההודעה בעברית "יום טוב לכולם" ונלחץ על Send ASCII:



במוניטור הטורי ב IDE של הארדוואינו נקבל :

The screenshot shows the Arduino IDE interface. The main window displays the source code for a program using the SerialBT library. The code includes comments in Hebrew and C++ code for initializing Bluetooth and sending a message. A serial monitor window is open, showing the output of the program. Two red arrows point from a text box to specific lines in the serial monitor output.

```
1 //This example code is in the Public Domain (or CC0 lic
2 //By Evandro Copercini - 2018
3 //
4 //This example creates a bridge between Serial and Clas
5 //and also demonstrate that SerialBT have the same func
6
7 #include "BluetoothSerial.h"
8
9 //#define USE_PIN // Uncomment this to use PIN during
10 const char *pin = "1234"; // Change this to more secur
11
12 String device_name = "ESP32-BT-Arik Porat";
13
14 #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUE
15 #error Bluetooth is not enabled! Please run `make menu
16 #endif
17
18 #if !defined(CONFIG_BT_SPP_ENABLED)
19 #error Serial Bluetooth not available or not enabled.
20 #endif
21
22 BluetoothSerial SerialBT;
23
24 void setup() {
```

Serial Monitor Output:

```
ets Jul 29 2019 12:21:46
rst:0x1 (POWERON_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:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0
The device with name "ESP32-BT-Arik Porat" is started.
Now you can pair it with Bluetooth!
Good day to everyone
יום טוב לכולם
```

Annotations:

- קיבלנו את ההודעות באנגלית (Received the messages in English) - points to "Good day to everyone"
- ובעברית (and in Hebrew) - points to "יום טוב לכולם"

www.arikporat.com