סריקת פס I2C

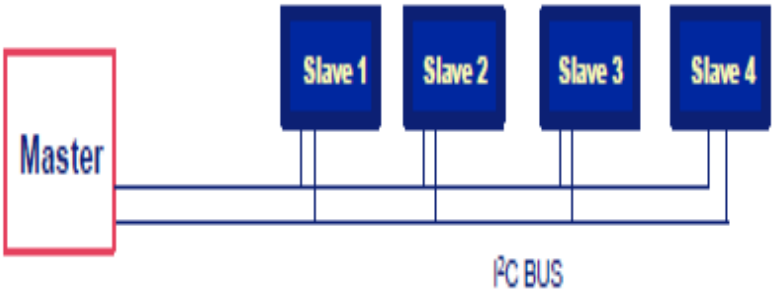
א. רקע תאורטי לתקשורת I2C

תקשורת I^2C נקרא גם IIC או I2C היא תקשורת טורית בין מעבד- מסטר - MASTER ורכיב עבד – SLAVE . הוא פותחה בתחילת שנות ה 80 על ידי חברת פיליפס הנקראת היום NXP semiconductors.

על פס תקשורת I^2C יכולים להתחבר מספר רכיבים שונים (זיכרונות , ממירים, שעוני זמן אמת , מרחיבי פס וכו'). הרכיב המנהל את תהליך התקשורת (המעבד) נקרא MASTER והרכיבים המתחברים אליו נקראים SLAVES . בתקשורת זו ישנם שני קווים. קו הנתונים הטורי – SDA (Serial Data) - שהוא דו כיווני וקו השעון הטורי - SCLK - (Serial CLocK) שהוא חד כיווני ומופעל על ידי ה MASTER .

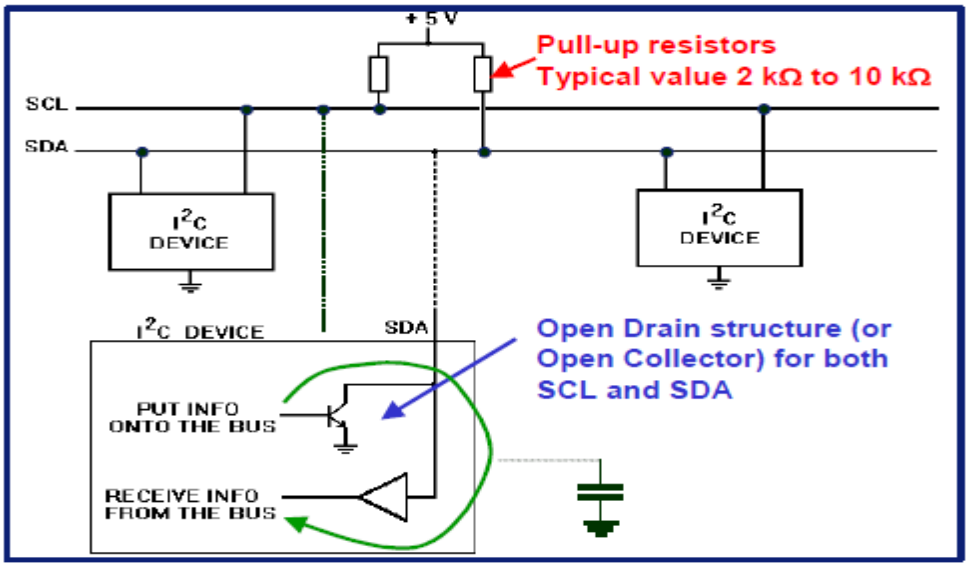
בנוסף, ה MASTER שולט על הגישה לפס ויוצר את מצבי ה START (התחלה) וה STOP (סיום).

איור 1 א' מתאר מספר רכיבים המתחברים על קו התקשורת I^2C



איור 1 - חיבור של מספר רכיבי SLAVE אל MASTER

באיור 1 ניתן לראות 4 רכיבי SLAVE המתחברים אל MASTER. באיור 2 יש פרוט של נגדי ה PullUp וכיצד נראית דרגת היציאה והכניסה של רכיב המתחבר בתקשורת I^2C .



איור 2 - קו תקשורת I^2C מפורט

ניתן לראות שעל 2 הקווים SDA (קו הנתון) ו SCL (קו השעון) יכולים להתחבר מספר רכיבים . לכל רכיב יש כתובת ייחודית משלו. לרכיב DS1307 הכתובת היא 68H (וכאשר מזיזים ביטים פעם אחת שמאלה – יוסבר מדוע בהמשך) הכתובת היא

1101000X (D0H לכתיבה או D1H לקריאה). לרכיב קול של חברת WINBOND הנקרא ISD5116 הכתובת היא 40H או 80H לאחר הזזה של ביט שמאלה וכך הלאה.

באיור רואים 2 רכיבים המתחברים על הקווים. בחלק התחתון של האיור רואים מבנה פנימי של רכיב ורואים שהרכיב מתחבר בעזרת חוצץ (מתואר על ידי המשולש) המקבל נתון מהקו. מעל החוצץ יש טרנזיסטור בחיבור קולט פתוח (Open Collector) או טרנזיסטור תופעת שדה - FET - בחיבור מפק פתוח (Open Drain) , שיכול לכתוב לקו נתון.

לטרנזיסטור יש לחבר נגד חיצוני שערכו נע בין 2 קילו אוהם ל 10 קילו אוהם. הערכים נבחרים כך שמצד אחד הנגדים לא יהיו קטנים מידי כדי שלא יזרום זרם גדול דרך הקווים ודרך הרכיב (במצב שהרכיב מוציא 0) ומצד שני שהנגד לא יהיה גדול מידי כי הוא קובע את זמן הטעינה והפריקה במעברים בין 0 ל 1 ולהפך ונגד גדול מידי יגביל את קצב התקשורת.

ד.1 כללים והגדרות בתקשורת I^2C

- העברה נתון יכולה להתחיל רק כאשר הקו לא עסוק - NOT BUSY .
- בזמן העברת נתון, קו הנתון חייב להישאר יציב כאשר קו השעון במצב גבוה. שינוי בקו הנתון כאשר קו השעון הוא גבוה יתפרש כאות בקרה.

מגדירים את מצבי הפס הבאים :

Bus Not Busy - פס לא עסוק

גם קו הנתון וגם קו השעון בגבוה.

START DATA TRANSFER - התחל העברת נתון

שינוי במצב קו הנתון מגבוה לנמוך כאשר השעון נמצא בגבוה מוגדר כמצב START .

STOP DATA TRANSFER - עצור העברת נתון

שינוי במצב קו הנתון מנמוך לגבוה כאשר השעון במצב גבוה מוגדר כמצב STOP .

DATA VALID - תקפות נתון

מצב קו הנתון מייצג תקפות נתון כאשר לאחר מצב START , קו הנתון יציב למשך זמן הגבוה של אות השעון. הנתון בקו חייב להשתנות רק בזמן מצב נמוך של אות השעון. יש פולס שעון אחד עבור כל ביט של נתון.

כל העברת נתונים מתחילה עם מצב START ומסתיימת עם מצב STOP . כמות הבתים המועברת בין START ל STOP לא מוגבלת ונקבעת על ידי רכיב ה MASTER . האינפורמציה מועברת ביית אחרי ביית וכל מקלט מאשר קבלת הבית עם ביט תשיעי של ACKNOWLEDGE .

בהגדרות של I^2C יש תקן של קצב ב 100KHz ויש תקן ל 400KHz .

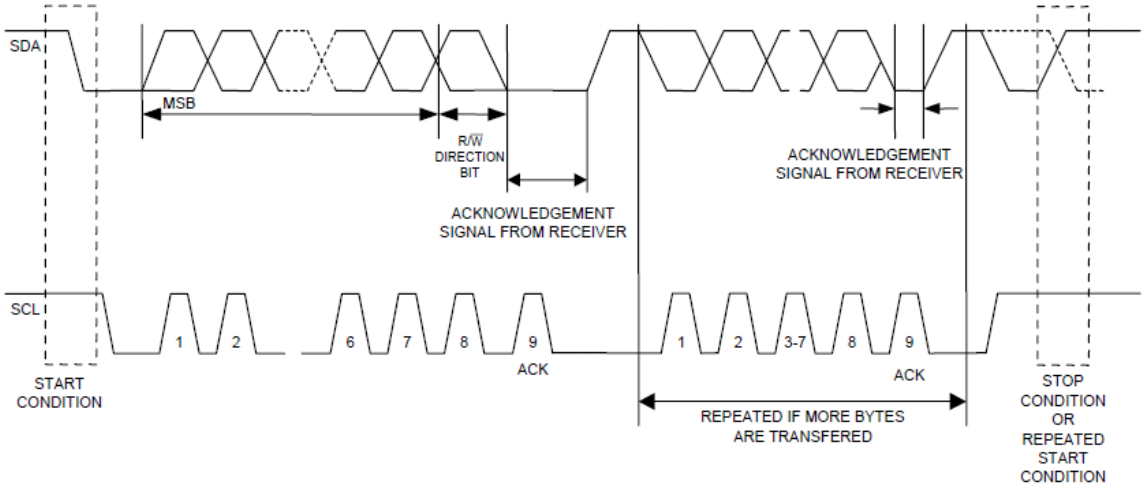
ACKNOWLEDGE – אישור

כל רכיב קולט חייב בסיום קליטת ביית, שהועבר אליו, ליצור ביט ACKNOLEDGE . רכיב הMASTER יוצר פולס שעון נוסף הקשור לביט זה.

רכיב היוצר ACKNOLEDGE חייב להוריד את קו הנתון הטורי – SDA – ל 0 בזמן פולס השעון, כלומר שקו הנתון יהיה יציב בנמוך בזמן שקו השעון בגבוה. רכיב ה MASTER מסמן ל SLAVE על סיום התקשורת על ידי **אי יצירת** ביט ה

ACKNOLEDGE כאשר הוא קלט את הביית האחרון מה SLAVE . במקרה כזה על ה SLAVE להשאיר את קו הנתון בגבוה כדי לאפשר ל MASTER ליצור מצב STOP .

באיור 3 ניתן לראות העברה של נתון טורי.



איור 3 - העברת נתון בקו תקשורת טורית I^2C

את הקו SCL (הקו התחתון בשרטוט) של פולסי השעון יוצר תמיד ה MASTER. כדאי לשים לב שמצב START קורה כאשר קו SCL בגבוה ואז ה MASTER הוריד את קו הנתון ל 0 . לאחר מכן ה MASTER יוצר 8 פולסי שעון ואז הוא שולח בקו הנתון - SDA - 8 ביטים . 7 ביטים הם כתובת הרכיב והביט ה 8 אומר האם הוא רוצה לכתוב אל הרכיב או לקרא ממנו (0 - כתיבה, 2 – קריאה). לאחר מכן ה MASTER יוצר פולס 9 נוסף שבו ה SLAVE צריך להחזיר ACKNOWLEDGE . לאחר מכן אין צורך ב START נוסף והבתים נשלחים אחד אחרי השני כאשר הצד הקולט נותן ACKNOWLEDGE בביט מספר 9 . מצב STOP (או START חוזר) מתואר בצד ימין של איור 6 . הוא נוצר כאשר קו השעון ב 1 ואז בקו הנתון יש מעבר מ 0 ל 1 . מצב START חוזר משורטט בקו מקווקו ובו רואים שבזמן שקו השעון ב 1 יורד קו הנתון ל 0 .

ד.2 העברת נתון בתקשורת I^2C

שתי אפשרויות העברת נתונים קיימות בקו תקשורת I^2C :

א. ה MASTER משדר וה SLAVE קולט - אופן כתיבה - Write Mode

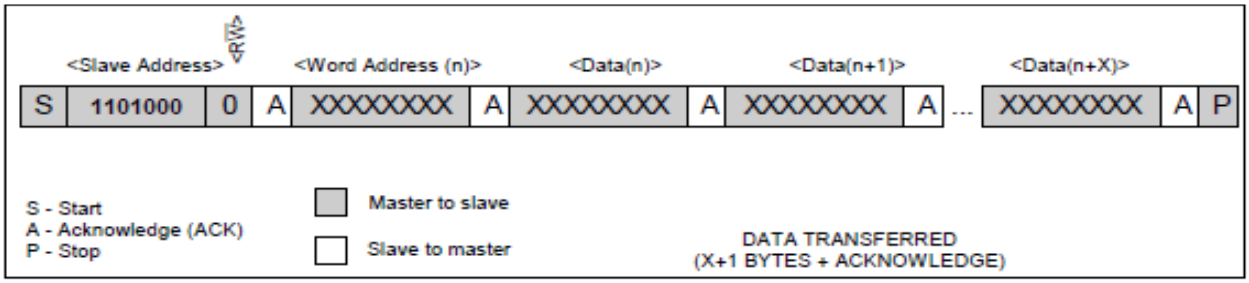
במקרה זה הביית הראשון המשודר על ידי ה MASTER הוא **הכתובת** של ה SLAVE המכילה 7 ביטים מביט D1 עד D7 ועוד ביט LSB (ביט D0) המציין האם מבצעים כתיבה (בביט יש 0) או קריאה (בביט יש 1). לדוגמה במקרה של רכיב DS1307 או DS3231 שהם רכיבי שעון זמן אמת – RTC - הכתובת היא 0110100**X** - 0x68 במקרה של **כתיבה** לרכיב או 0x69 אם **קוראים** מהרכיב . לאחר מכן יבואו מספר בתים של נתונים. ה SLAVE מחזיר ACKNOLEDGE בסיום כל ביית נתונים שקלט. העברת הנתון מתחילה **מביט ה MSB** ראשון ועד ביט ה **LSB** !!

ב. ביית משודר מה SLAVE אל ה MASTER - אופן קריאה - Read Mode

במקרה זה הביית הראשון שנשלח הוא על ידי ה MASTER השולח את כתובת ה SLAVE שמחזיר מצידו את ביט ה ACKNOWLEDGE . מכאן ה SLAVE שולח מספר בתי נתונים. ה MASTER מחזיר ביט ACKNOLEDGE אחרי כל

קליטת ביית נתון חוץ מהביית האחרון שהוא איננו מחזיר ACNOWLEDGE או אפשר להגיד שהוא מחזיר Not ACKNOWLEDGE .

ד.3 אופן כתיבה – ה MASTER משדר אל אחד מה SLAVES



איור 4 - אופן כתיבת נתון מה MASTER כשה SLAVE הוא המקלט .

באיור 4 מתואר מצב שבו ה MASTER כותב אל ה SLAVE המשמש כמקלט. החלק הכהה שבאיור הוא מה ששולח ה MASTER. החלק הבהיר הוא המסומן כ A הוא ה Acknowledge – אישור - ששולח ה SLAVE .

ה MASTER יוצר מצב START (מסומן ב S). לאחר מכן הוא שולח 7 ביטים של כתובת הרכיב - 68H במקרה של הרכיב DS1307 - והביט ה 8 הוא 0 המציין שהוא הכותב וה SLAVE הוא המקלט. על ה SLAVE לענות ב ACKNOLEDGE (מסומן ב A). לאחר מכן ה MASTER שולח ביית נוסף הטוען את מצביע (אוגר) הכתובות (במקרה שברכיב יש זיכרון או יש קבוצת רגיסטרים הנמצאים בכתובות עוקבות) בתוך הרכיב. הביית השלישי הוא כבר נתון הנכתב לכתובת זו ומצביע הכתובות גדל אוטומטית ב 1. כל נתון נכתב בכתובת שבמצביע הכתובות ומצביע הכתובות מתקדם ב 1 . אחרי כל ביית שנקלט על ידי ה SLAVE הוא שולח אישור – ACKNOELEDGE . ה MASTER מסיים את התקשורת בעזרת מצב STOP (מופיע בצד ימין עם האות P).

ד.4 אופן קריאה - ה SLAVE משדר אל ה MASTER

גם מצב זה מתחיל תמיד במצב שבו ה MASTER משדר אל ה SLAVE אבל כאן הוא אומר שהוא רוצה לקרא ממנו (הוא שם בביט ה LSB של הכתובת 1) . הביית הראשון שה MASTER משדר נקלט על ידי ה SLAVE כמו שתואר בפסקה הקודמת, כלומר ה MASTER ייצור מצב START , יישלח את 7 הביטים של הכתובת **1**0110001 אבל בביט השמיני – LSB - יהיה 1 שבו הוא אומר שהוא רוצה לקרא. מכאן ה SLAVE משדר את הנתונים וה MASTER עונה עם ביט ACKNOLEDGE . בביית האחרון , כשה MASTER רוצה בסיום התקשורת , הוא איננו מגיב בביט ה 9 ולא שולח ACKNOLEDGE (מסומן באיור ב **Â** – not Acknoledge) וגם שולח ביט עשירי של STOP .

כאשר רוצים לקרוא מרכיב SLAVE שיש לו זיכרון שבהם יש מספר נתונים אז ה MASTER כותב 3 בתים אל הרכיב.

בביית הראשון ה MASTER אומר לרכיב ה SLAVE שהוא פונה אליו לכתיבה (בביט ה LSB שמים 0) .

ה SLAVE נותן **A** – אישור .

בביית השני הוא מציין את הכתובת בזיכרון ה SLAVE ממנה הוא רוצה להתחיל לקרוא . זוהי הכתובת שבזיכרון ה SLAVE והיא של 8 ביטים. פעולה זו באה לטעון ב SLAVE רגיסטר מונה כתובות (שנקרא גם מצביע כתובות) .

ה SLAVE נותן **A** – אישור .

מיד לאחר מכן ה MASTER שולח START נוסף.

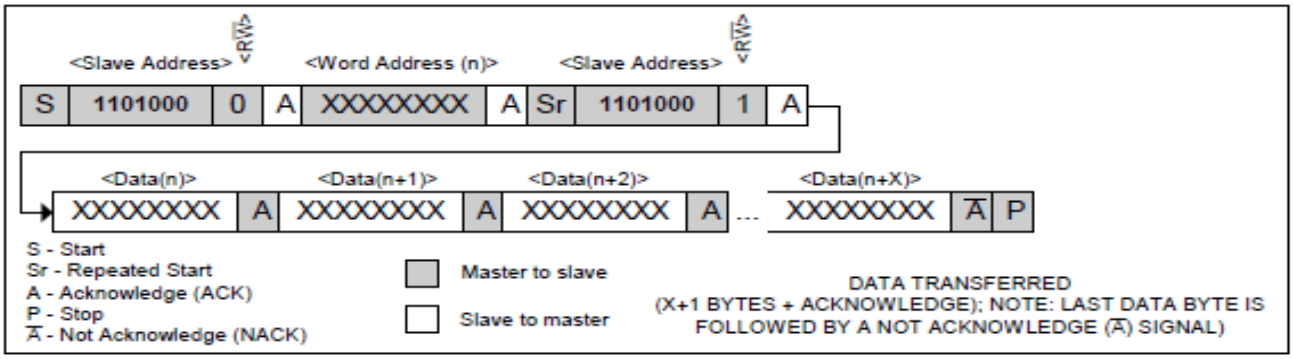
בביית השלישי ה MASTER שולח את כתובת ה SLAVE אבל בביט ה LSB יש 1 לציין קריאה .

ה SLAVE נותן אישור – **A** .

ה MASTER הופך את קו הנתון הטורי SDA לקלט.

מכאן ה SLAVE מתחיל לשלוח את הנתון מהכתובת שקיבל בבית השני וכל אישור - **A** מה MASTER יגרום למונה הכתובות שלו להתקדם ולשלוח את הנתון מהכתובת הבאה עד שלא יקבל אישור מה MASTER ולאחריו **S** המציין Stop – עצירה ואז התקשורת מסתיימת. מונה הכתובות ב SLAVE נשאר על הכתובת של הנתון הבא.

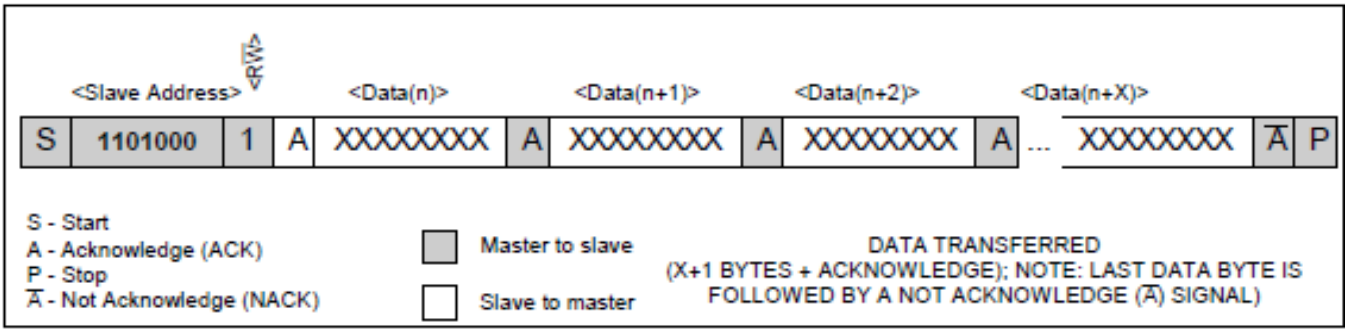
איור 5 מתאר פעולת קריאה מהכתובת הרצויה. גם כאן הצבע הכהה הוא של ה MASTER והבהיר של ה SLAVE .



איור 5 – פעולה משולבת של כתיבה ל SLAVE כדי לציין כתובת רצויה וקריאה מה SLAVE

מהאיור רואים שה MASTER שולח 3 בתים אל ה SLAVE . בית ראשון הוא כתובת רכיב ה SLAVE בפס I2C , הבית השני הוא הכתובת בזיכרון של ה SLAVE ממנה ה MASTER רוצה לקבל נתון. מיד לאחר מכן ה MASTER יצר מצב START חוזר (מסומן ב Sr), שלח בית נוסף שבו יש את כתובת הרכיב 01101001 (הביט השמיני הוא של קריאה) ומרגע זה ה SLAVE משדר וה MASTER מגיב ב ACKNOLEDGE . בסיום התקשורת ה MASTER לא נותן ACKNOLEDGE (מסומן ב **Â**) ולאחר מכן נותן מצב STOP . הנתונים המשודרים מה SLAVE מתחילים מהכתובת ששלח אליו ה MASTER . אם ה MASTER לא שולח כתובת אז הנתון שמתקבל יהיה מהכתובת האחרונה שבה נמצא מצביע הכתובות של ה SLAVE. כל נתון שה SLAVE שולח הוא מקדם את מצביע הכתובות שלו לכתובת הבאה ושולח את הנתון מכתובת זו וכל עוד הוא מקבל אישור **A** Acknowledge מהמסטר הוא משדר את הנתון הבא.

איור 6 מתאר מצב תקשורת זה. גם כאן הצבע הכהה הוא של ה MASTER והבהיר של ה SLAVE .



איור 6 - אופן קריאה ללא ציון כתובת ב SLAVE.

מהאיור רואים שה MASTER שולח מצב START ומיד אחריו ביית שמציין את כתובת ה SLAVE בפס I2C עם 1 בביט ה LSB המציין קריאה. מרגע זה ה MASTER הופך את קו הנתון SDA לקלט וקולט את הביית ששולח ה SLAVE . כל נתון

הנקלט על ידי ה MASTER נענה ב A – אישור של ה MASTER ואז ה SLAVE שולח ביית חדש. כאשר ה MASTER רוצה סיום תקשורת הוא לא נותן אישור – מצב **Â** ואז שולח מצב **Stop** והתקשורת מסתיימת.

ב. I2C במיקרו בקר ESP32

ל-ESP32 יש שני ממשקי פס I2C שיכולים לשמש כאדון או עבד I2C. נסביר כאן את פרוטוקול התקשורת של I2C של ה-ESP32 באמצעות Arduino IDE . כיצד לבחור פיני I2C , לחבר התקני I2C מרובים לאותו פס וכיצד להשתמש בשני ממשקי הפס של I2C .

ה-ESP32 תומך בתקשורת I2C באמצעות שני ממשקי פס ה-I2C שלו שיכולים לשמש כמסטר או כעבד בהתאם לתצורת המשתמש. בהתאם לדף הנתונים של ESP32 ממשקי ה-I2C של ESP32 תומכים ב:

- מצב רגיל (Kbit/s 100)
- מצב מהיר (Kbit/s 400)
- עד 5 מגה-הרץ, אך מוגבל על-ידי עוצמת המשיכה של SDA (נגד ה Pull UP והקיבול של הקו).
- מיעון כתובות - מצב פנייה לכתובות - של 7 סיביות/10 סיביות.
- מצב ממוען כפול. ניתן לתכנת את אוגרי הפקודות גם ל 7 וגם ל 10כך שתהיה להם גמישות רבה יותר.

קווי ה-SDA וה-SCL פעילים בנמוך ולכן יש למשוך אותם למעלה -pull up - באמצעות נגדים. ערכים אופייניים הם 4.7k Ohm עבור מכשירי 5V ו- 2.4k Ohm עבור התקני 3.3V.

רוב החיישנים שאנו משתמשים בהם בפרויקטים שלנו הם מודולים שכבר יש להם את נגדי ה pull up מובנים במודול. לכן, בדרך כלל לא צריך לדאוג לגבי זה.

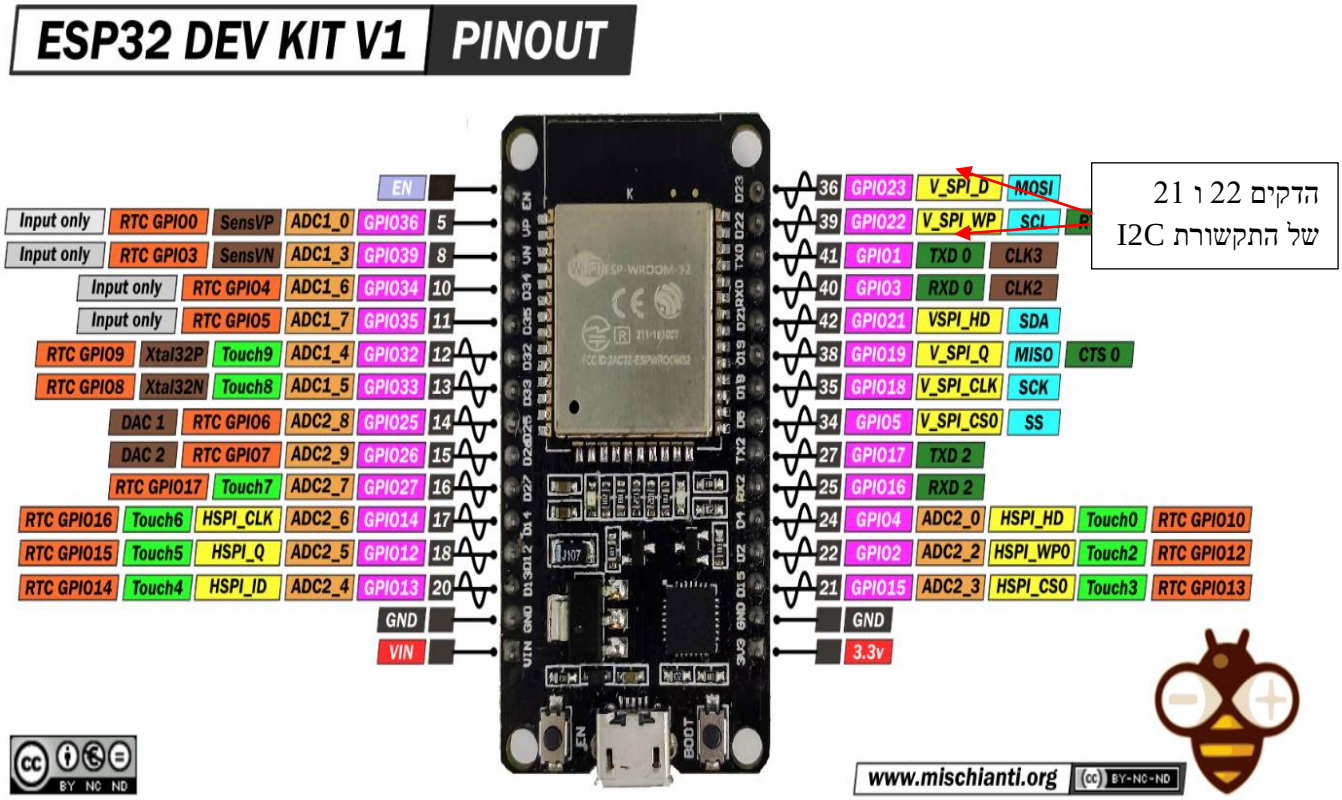
חיבור התקן I2C ל-ESP32 הוא בדרך כלל פשוט. יש לחבר 4 קווים : קו האדמה GND ל GND , קו הנתון הטורי SDA ל-SDA , קו פולסי השעון SDA, ל SCL וקו ספק כוח חיובי לציוד היקפי שהוא בדרך כלל 3.3 וולט (אבל זה תלוי במודול שבו נשתמש).

מספר ההדק ב ESP32	רכיב/מודול I2C
SDA (default is GPIO 21)	SDA
SCL (default is GPIO 22)	SCL
GND	GND
בדרך כלל 3.3V או 5V	Vcc

טבלה 1 : חיבור מודול I2C אל ESP32

בעת שימוש ב- ESP32 עם Arduino IDE הדקי ה- I2C המוגדרים כברירת מחדל הם (SCL) GPIO 22 ו- (SDA) GPIO 21, אך ניתן להגדיר בתוכנית שלנו לשימוש בכל הדק אחר.

האיור הבא מתאר את הדקי ה ESP32 בכרטיס ESP32 DEV KIT V1

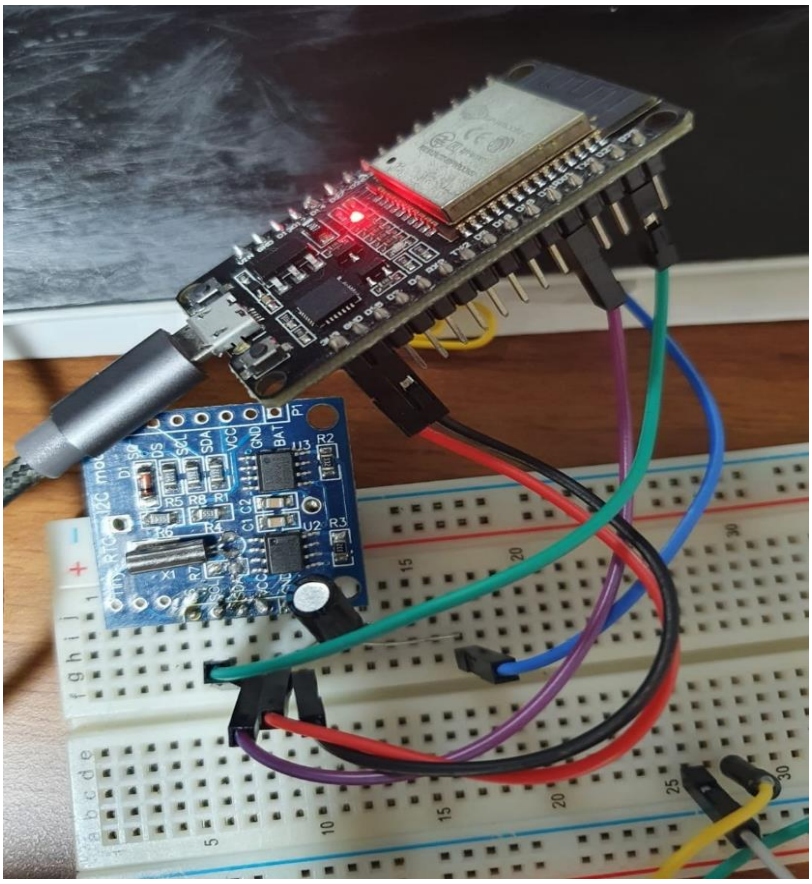


איור 7 : הדקי התקשורת SDA ו SCL של תקשורת I2C ברכיב ESP32 DEV KIT V1

ג. סריקת כתובת עם הדקי I2C של ה ESP32

בתקשורת I2C לכל SLAVE בפס יש כתובת משלו, מספר הקסדצימלי המאפשר ל-ESP32 לתקשר עם כל מכשיר.

חיברנו רכיב RTC – שעון זמן אמת – DS3231 אל הדקי התקשורת I2C של הניקרו בקר ESP32 . החיבור הוא של 4 הדקים ונראה באיור הבא . באיור רואים גם חוט בצבע כחול המחובר להדק EN של ה ESP32 . שמתי קבל של 10 מיקרו פאראד בין הדק ה EN לאדמה כי מידי פעם בהעלאת התוכנית לא הייתה מתבצעת הורדה. הוספת הקבל תיקנה את הבעייה.



איור 8 : חיבור DS3231 אל ESP32 בעזרת מגשרי קצר – Jumpers .

את כתובת I2C ניתן למצוא בדרך כלל בגליון הנתונים של הרכיב. עם זאת כדי לגלות את כתובת הרכיב ושהרכיב אכן מחובר נכון ופרוטוקול ה I2C שלו פועל נפעיל את התוכנית סורק I2C הבאה שנכתבה על ידי Rui Santos באתר של randomnerd :

```
/*********
  Rui Santos
  Complete project details at https://randomnerdtutorials.com
  התוכנית בודקת האם יש רכיבים המתחברים בתקשורת I2C
  בעזרת הדקי התקשורת של המיקרו. הדק 22 הוא הדק השעון SCL
  והדק 21 הוא הדק הנתון הטורי SDA
*********/
```

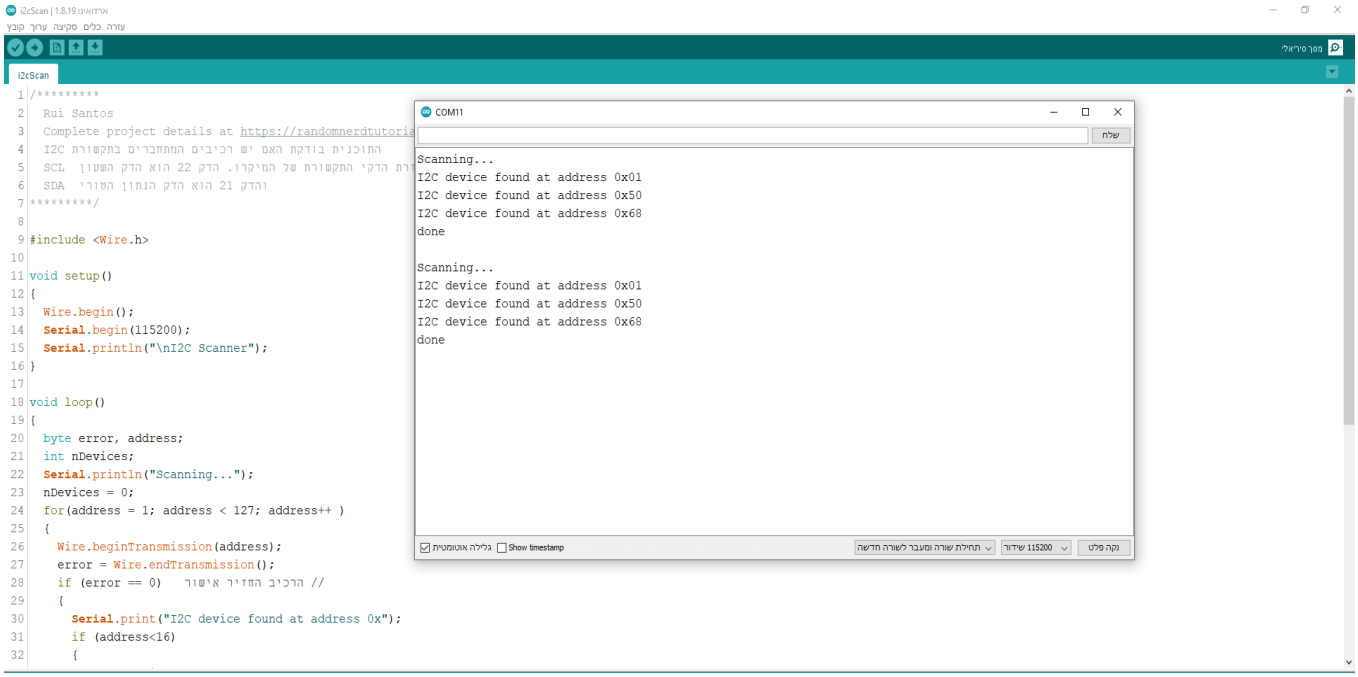
```
#include <Wire.h>

void setup()
{
  Wire.begin();
  Serial.begin(115200);
  Serial.println("\nI2C Scanner");
}

void loop()
{
  byte error, address;
  int nDevices;
  Serial.println("Scanning...");
  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    if (error == 0)  // הרכיב החזיר אישור
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
      {
        Serial.print("0");
      }
      Serial.println(address,HEX);
      nDevices++;
```

```
    }
    else if (error==4)
    {
      Serial.print("Unknow error at address 0x");
      if (address<16) {
        Serial.print("0");
      }
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
  {
    Serial.println("No I2C devices found\n");
  }
  else {
    Serial.println("done\n");
  }
  delay(5000);
}
```

התוצאה שנקבל במסך הטורי נראית באיור הבא :



איור 10 : התוצאה המתקבלת במסך הטורי בהרצת התוכנית

ניתן לראות שהרכיב "יושב" על 3 כתובות של I2C , שהן 0x01, 0x50 0x68 . לכתובת 0x68 ציפינו כי זוהי כתובתו הידועה של ה RTC . מדוע קיבלנו 2 כתובות נוספות – שווה חקירה....

ד. סריקת כתובת I2C עם 2 הדקים כלליים

בפרק זה נרשום תוכנית המשתמשת ב 2 הדקי GPIO שנבחר המתחברים ל SDA ו SCL של המודול. אם בתוכנית הקודמת השתמשנו בספריות מוכנות שלא אנחנו כתבנו , בתוכנית זו נרשום את הפונקציות של פרוטוקול I2C בתוך התוכנית ללא #include לספריות מוכנות , כך שהתלמיד/סטודנט יוכל להבין לעומק את הפרוטוקול.

בתוכנית הוספנו מצב שכאשר מזוהה רכיב באחת הכתובות מקבלים במסך הטורי של ה IDE את הרכיב/רכיבים שיכולים להיות בכתובות אלו.

עוד תוספת בתוכנית הוא משתנה בוליאני הנקרא debug . אם נשים בו 1 מקבלים במסך הטורי את הכתובת הנוכחית שהתוכנית סורקת והאם נמצא בה רכיב I2C . במידה וכן נכתב למסך הטורי מיהו הרכיב/רכיבים שבכתובת זו. כאשר נשים בביט 0 התוכנית מבצעת סריקה ורק כאשר מתגלה רכיב בכתובת הנסרקת הוא נרשם.

החומרה שנחבר היא : את הדק ה SDA של ה RTC נחבר להדק GPIO15 של ה ESP32 ואת הדק SCL של המודול נחבר אל הדק GPIO4 של ה ESP32 . את הדק 3.3 וולט של ה RTC נחבר להדק 3.3 וולט בכרטיס ה ESP32 ואת ה GND של RTC להדק GND של ה ESP32 .

כפי שצויין הפונקציות start( ) , stop ( ) , ו ack רשומות בתוכנית עצמה.

כמו כן נמצאת הפונקציה void sendByte(byte data8) המקבלת ביית של 8 ביטים ומוציאה אותו אל קו ה SDA ביט אחרי ביט החל מביט ה MSB ועד לביט ה LSB .

התוכנית נראית כך :

```
// ---------------------------------------
//התכנית סורקת את הכתובות מ 0 ועד 0x7f ובודקת האם רכיב I2C נמצא ובאיזו כתובת
#define sda 15
#define scl 4
byte  address;  //  כתובת הרכיב
byte error;  // מראה שיש שגיאה
byte numberOfDevicesFound;
bool debug=1; // מראה האם להציג את הכתובת שבה לא נמצא רכיב . אם שמים 0 רואים רק את הכתובת שבה נמצא רכיב
String found=" Adreses were Found";
//---------- מערך מחרוזות של כתובות ומהו הרכיב/ים שנמצאים בכתובת הזו ----------
String  deviceAddress[256]={
//const char devicesAddresses[100][256]PROGMEM = {
"0x0 Reserved",
"0x1 Reserved",
"0x2 Reserved",
"0x3 Reserved",
"0x4 Reserved",
"0x5 Reserved",
"0x6 Reserved",
"0x7 Reserved",
"0x8 Unknown Device",
"0x9 Unknown Device",
"0xa Unknown Device",
"0xb Unknown Device",
"0x0c AK8975 - 3-axis magnetometer",
"0x0d AK8975 - 3-axis magnetometer",
"0x0e AK8975 or IST-8310 or MAG3110 - 3-axis magnetometer",
"0x0f AK8975 - 3-axis magnetometer",
"0x10 VEML7700 or VML6075 or VEML6075 - Light Sensor",
"0x11 Si4713 SAA5243-Computer controlled teletext circuit or SAA5246 - Integrated VIP and teletext or
Si4713 - FM Radio Transmitter with Receive Power Scan",
"0x12 SEN-17374 - Sparkfun EKMC4607112K PIR",
"0x13 SEN-17374 - Sparkfun EKMC4607112K PIR or VCNL40x0 - proximity sensor",
```

"0x14 Unknown Device",
"0x15 Unknown Device",
"0x16 Unknown Device",
"0x17 Unknown Device",
"0x18 COM-15093 or LIS3DH or LSM303 - 3 axis accelrometer or MCP9808 - Digital Temperature Sensor",
"0x19 MCP9808 LIS3DH LSM303 COM-15093",
"0x1a MCP9808 Temperature Sensor",
"0x1b MCP9808 Temperature Sensor",
"0x1c MCP9808 Temperature Sensor - MMA845x 3-axis, 14-bit/8-bit digital accelerometer FXOS8700 - 6-axis sensor with integrated linear accelerometer and magnetometer",
"0x1d MCP9808 Digital Temperature Sensor, MMA845x 3-axis, 14-bit/8-bit digital acceleromete, ADXL345 3-axis accelerometer, FXOS8700 6-axis sensor ...",
"0x1e HMC5883 LSM303 MCP9808 LSM303 FXOS8700",
"0x1f MCP9808 FXOS8700",
"0x20 TCA9554 MCP23008 MA12070P MCP23017 Chirp! FXAS21002",
"0x21 FXAS21002 MCP23008 MCP23017 SAA4700 MA12070P TCA9554",
"0x22 MCP23008 MCP23017 PCA1070 MA12070P TCA9554",
"0x23 MCP23008 MCP23017 SAA4700 MA12070P TCA9554",
"0x24 TCA9554 MCP23008 PCD3312C MCP23017 PCD3311C",
"0x25 TCA9554 MCP23008 PCD3312C MCP23017 PCD3311C",
"0x26 MCP23008 MCP23017 TCA9554",
"0x27 MCP23008 MCP23017 HIH6130 TCA9554",
"0x28 BNO055 CAP1188",
"0x29 BNO055 VL53L0x VL6180X CAP1188 TCS34725 TSL2591",
"0x2a CAP1188 8-channel Capacitive Touch",
"0x2b CAP1188 8-channel Capacitive Touch",
"0x2c CAP1188 8-channel Capacitive Touch, AD5248 Digital Potentiometer, AD5251 Dual 64-Position I2C Nonvolatile Memory Digital Potentiometers, AD5252 Dual 256-Position I2C Nonvolatile Memory Digital Potentiometers, CAT5171",
"0x2d CAP1188 8-channel Capacitive Touch, AD5248 Digital Potentiometer, AD5251 Dual 64-Position I2C Nonvolatile Memory Digital Potentiometers, AD5252 Dual 256-Position I2C Nonvolatile Memory Digital Potentiometers, CAT5171",

"0x2e AD5248 Digital Potentiometer, AD5251 Dual 64-Position I2 C Nonvolatile Memory Digital Potentiometers, AD5252 Dual 256-Position I2C Nonvolatile Memory Digital Potentiometers, LPS22HB",
"0x2f AD5248 Digital Potentiometer, AD5243 AD5251 Dual 64-Position I2 C Nonvolatile Memory Digital Potentiometers, AD5252 Dual 256-Position I2C Nonvolatile Memory Digital Potentiometers",
"0x30 SAA2502 MPEG audio source decoder",
"0x31 SAA2502 MPEG audio source decoder",
"0x32 Unknown Device",
"0x33 MLX90640 far infrared thermal sensor array (32x24 RES)",
"0x34 Unknown Device",
"0x35 Unknown Device",
"0x36 Unknown Device",
"0x37 Unknown Device",
"0x38 FT6x06 Capacitive Touch Driver, VEML6070 UVA Light Sensor, BMA150 triaxial acceleration sensor, SAA1064 4-digit LED driver, SEN-15892 Zio Qwiic Loudness Sensor, PCF8574AP",
"0x39 TSL2561 light sensor, APDS-9960 IR/Color/Proximity Sensor, VEML6070 UVA Light Sensor, SAA1064 4-digit LED driver, PCF8574AP I²C-bus to parallel port expander",
"0x3a PCF8577C SAA1064 PCF8574AP I²C-bus to parallel port expander",
"0x3b SAA1064 4-digit LED driver, PCF8569 LCD column driver for dot matrix displays, PCF8574AP I²C-bus to parallel port expander",
"0x3c SSD1305 132 x 64 Dot Matrix OLED/PLED, SSD1306 128 x 64 Dot Matrix Monochrome OLED/PLED, PCF8578 Row/column LCD dot matrix driver/display, PCF8569 LCD column driver for dot matrix displays, SH1106 PCF8574AP I²C-bus to parallel port expander",
"0x3d SSD1305 132 x 64 Dot Matrix OLED/PLED, SSD1306 128 x 64 Dot Matrix Monochrome OLED/PLED, PCF8578 Row/column LCD dot matrix driver/display, SH1106 132 X 64 Dot Matrix OLED/PLED, PCF8574AP I²C-bus to parallel port expander",
"0x3e PCF8574AP I²C-bus to parallel port expander",
"0x3f PCF8574AP I²C-bus to parallel port expander",
"0x40 Si7021/HTU21D-F Humidity/Temp sensor, TMP007 IR Temperature sensor, TMP006 Infrared Thermopile Sensor, PCA9685 16-channel PWM driver default address, INA219 26V Bi-Directional High-Side Current/Power/Voltage Monitor, TEA6330 Sound fader control circuit for car radios, TEA6300 Sound fader control and preamplifier/source selector, TDA9860 Hi-fi audio processor, TEA6320 4-input tone/volume controller with fader control, TDA8421 Audio processor, NE5751 Audio processor, INA260 Precision Digital Current and Power Monitor, PCF8574 Remote 8-Bit I/O Expander",

"0x41 TMP007 IR Temperature sensor, TDA8421 Audio processor, STMPE610 Resistive Touch controller, TDA8424 Audio processor, STMPE811 Resistive touchscreen controller, PCF8574 Remote 8-Bit I/O Expander, NE5751 Audio processor, INA260 Precision Digital Current and Power Monitor, TDA8425 Audio processor, TMP006 Infrared Thermopile Sensor, TDA9860 Hi-fi audio processor, INA219 26V Bi-Directional High-Side Current/Power/Voltage Monitor, PCA9685 16-channel PWM driver default address, TDA8426 Hi-fi stereo audio processor",
"0x42 TMP007 IR Temperature sensor, TDA8417 HDC1008 PCF8574 INA260 TDA8415 TMP006 INA219 PCA9685",
"0x43 TMP007 IR Temperature sensor, HDC1008 PCF8574 INA260 TMP006 INA219 PCA9685",
"0x44 TMP007 IR Temperature sensor, TMP006 PCA9685 INA219 STMPE610 SHT31 ISL29125 STMPE811 TDA4688 TDA4672 TDA4780 TDA4670 TDA8442 TDA4687 TDA4671 TDA4680 INA260 PCF8574",
"0x45 TMP007 IR Temperature sensor, TDA7433 TDA8376 PCF8574 INA260 TMP006 INA219 PCA9685 SHT31",
"0x46 TMP007 IR Temperature sensor, PCF8574 TDA8370 INA260 TMP006 INA219 PCA9685 TDA9150",
"0x47 TMP007 IR Temperature sensor, PCF8574 INA260 TMP006 INA219 PCA9685",
"0x48 PCA9685 16-channel PWM driver default address, INA219 26V Bi-Directional High-Side Current/Power/Voltage Monitor, PN532 NFC/RFID reader, TMP102 Temperature sensor, INA260 Digital Current and Power Monitor, ADS1115 4-channel 16-bit ADC, PCF8574 I²C-bus to parallel port expander, ADS7828 12-Bit, 8-Channel Sampling ANALOG-TO-DIGITAL CONVERTER",
"0x49 TSL2561 PCA9685 16-channel PWM driver default address, INA219 26V Bi-Directional High-Side Current/Power/Voltage Monitor, TMP102 Temperature sensor, INA260 Digital Current and Power Monitor, ADS1115 4-channel 16-bit ADC, AS7262 6-channel visible spectral_ID device, PCF8574 I²C-bus to parallel port expander, ADS7828",
"0x4a ADS7828 PCF8574 I²C-bus to parallel port expander, ADS1115 INA260 MAX44009 INA219 PCA9685 TMP102",
"0x4b ADS7828 PCF8574 I²C-bus to parallel port expander, ADS1115 INA260 MAX44009 INA219 PCA9685 TMP102",
"0x4c PCA9685 16-channel PWM driver default address, INA219 26V Bi-Directional High-Side Current/Power/Voltage Monitor, INA260 Digital Current and Power Monitor, PCF8574 I²C-bus to parallel port expander",

"0x4d PCA9685 16-channel PWM driver default address, INA219 26V Bi-Directional High-Side Current/Power/Voltage Monitor, INA260 Digital Current and Power Monitor, PCF8574 I²C-bus to parallel port expander",
"0x4e PCA9685 16-channel PWM driver default address, INA219 26V Bi-Directional High-Side Current/Power/Voltage Monitor, INA260 Digital Current and Power Monitor, PCF8574 I²C-bus to parallel port expander",
"0x4f PCA9685 16-channel PWM driver default address, INA219 26V Bi-Directional High-Side Current/Power/Voltage Monitor, INA260 Digital Current and Power Monitor, PCF8574 I²C-bus to parallel port expander",
"0x50 PCA9685 16-channel PWM driver default address, MB85RC - Ferroelectric RAM",
"0x51 PCA9685 16-channel PWM driver default address, MB85RC - Ferroelectric RAM VCNL4200 Proximity and Ambient Light Sensor",
"0x52 PCA9685 16-channel PWM driver default address, MB85RC Nunchuck controller APDS-9250 SI1133",
"0x53 ADXL345 PCA9685, 16-channel PWM driver default address, MB85RC - Ferroelectric RAM, Qwiic EEPROM - 512Kbit - 64KB",
"0x54 PCA9685 16-channel PWM driver default address, MB85RC - Ferroelectric RAM",
"0x55 PCA9685 16-channel PWM driver default address, MB85RC - Ferroelectric RAM, MAX30101 SI1133",
"0x56 PCA9685 16-channel PWM driver default address, MB85RC - Ferroelectric RAM",
"0x57 PCA9685 16-channel PWM driver default address, MB85RC - Ferroelectric RAM, MAX3010x - Pulse & Oximetry sensor ",
"0x58 PCA9685 16-channel PWM driver default address TPA2016 SGP30",
"0x59 PCA9685 16-channel PWM driver default address",
"0x5a MPR121 12-point capacitive touch sensor, MLX90614 -IR temperature sensor, CCS811 PCA9685 DRV2605 Haptic Motor Driver",
"0x5b PCA9685 16-channel PWM driver default address, CCS811 MPR121",
"0x5c PCA9685 16-channel PWM driver default address, AM2315 MPR121",
"0x5d PCA9685 16-channel PWM driver default address, MPR121 - 12-point capacitive touch sensor",
"0x5e PCA9685 16-channel PWM driver default address",
"0x5f PCA9685 16-channel PWM driver default address, HTS221",
"0x60 SI1132 Light Sensor, Si5351A Clock Generator, ATECC608A Microchip CryptoAuthentication™ Device, TSA5511 1.3 GHz PLL frequency synthesizer for TV, ATECC508A Crypto Element, MCP4725A0 12-Bit Digital-to-Analog Converter with EEPROM Memory, SAB3035 Digital tuning circuit

for computer-controlled TV, SAB3037 Digital tuning circuit for computer-controlled TV, PCA9685 16-channel PWM driver default address, MCP4725A1 12-bit DAC, TEA5767 Radio receiver, MPL3115A2 Barometric Pressure, MPL115A2 Barometric Pressure, Si1145 Light Sensor",
"0x61 Si5351A Clock Generator, TSA5511 1.3 GHz PLL frequency synthesizer for TV, MCP4725A0 12-Bit Digital-to-Analog Converter, SAB3035/SAB3037 Digital tuning circuit for computer-controlled TV, TEA6100 FM/IF for computer-controlled radio, PCA9685 16-channel PWM driver default address, MCP4725A1 12-Bit Digital-to-Analog Converter",
"0x62 SCD40-D-R2 TSA5511 UMA1014T SAB3035 SAB3037 PCA9685 MCP4725A1",
"0x63 Si4713 TSA5511 UMA1014T SAB3035 SAB3037 PCA9685 MCP4725A1",
"0x64 PCA9685 16-channel PWM driver default address, MCP4725A2 12-Bit Digital-to-Analog Converter, MCP4725A1 12-Bit Digital-to-Analog Converter",
"0x65 PCA9685 16-channel PWM driver default address, MCP4725A2 12-Bit Digital-to-Analog Converter, MCP4725A1 12-Bit Digital-to-Analog Converter",
"0x66 PCA9685 16-channel PWM driver default address, MCP4725A3 12-Bit Digital-to-Analog Converter, IS31FL3731 144-LED Audio Modulated Matrix LED Driver (CharliePlex), MCP4725A1 12-Bit Digital-to-Analog Converter",
"0x67 PCA9685 16-channel PWM driver default address, MCP4725A3 12-Bit Digital-to-Analog Converter, MCP4725A1 12-Bit Digital-to-Analog Converter",
"0x68 MPU-9250 9-DoF IMU Gyroscope, Accelerometer and Magnetometer, ICM-20948 9-Axis Motion Tracking device, MPU6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Devices, DS3231 RTC, AMG8833 IR Thermal Camera Breakout, PCA9685 16-channel PWM driver default address, PCF8573 Clock/calendar Detector, PCF8523 RTC, DS1307 RTC, ITG3200 Gyro",
"0x69 MPU-9250 9-DoF IMU Gyroscope, Accelerometer and Magnetometer, ICM-20948 9-Axis Motion Tracking device, MPU6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Devices, AMG8833 IR Thermal Camera Breakout, PCA9685 16-channel PWM driver default address, PCF8573 ITG3200 SPS30",
"0x6a PCA9685 16-channel PWM driver default address, L3GD20H gyroscope, PCF8573 Clock/calendar with Power Fail Detector",
"0x6b PCA9685 16-channel PWM driver default address, L3GD20H gyroscope, PCF8573 Clock/calendar with Power Fail Detector",
"0x6c PCA9685 16-channel PWM driver default address",
"0x6d PCA9685 16-channel PWM driver default address",
"0x6e PCA9685 16-channel PWM driver default address",

"0x6f PCA9685 16-channel PWM driver default address, MCP7940N Battery-Backed I2C Real-Time Clock/Calendar with SRAM",
"0x70 PCA9685 16-channel PWM driver default address, TCA9548 1-to-8 I2C Multiplexer, HT16K33 LED Matrix Driver, SHTC3 Humidity & Temperature Sensor",
"0x71 PCA9685 16-channel PWM driver default address, TCA9548 1-to-8 I2C Multiplexer, HT16K33 LED Matrix Driver",
"0x72 PCA9685 16-channel PWM driver default address, TCA9548 1-to-8 I2C Multiplexer, HT16K33 LED Matrix Driver",
"0x73 PCA9685 16-channel PWM driver default address, TCA9548 1-to-8 I2C Multiplexer, HT16K33 LED Matrix Driver",
"0x74 PCA9685 16-channel PWM driver default address, TCA9548 1-to-8 I2C Multiplexer, HT16K33 LED Matrix Driver",
"0x75 PCA9685 16-channel PWM driver default address, TCA9548 1-to-8 I2C Multiplexer, HT16K33 LED Matrix Driver",
"0x76 BME688 low power gas, pressure, temperature and humidity sensor, BME680 Low power gas, pressure, temperature & humidity sensor, MS5611 Barometric Pressure, MS5607 Barometric Pressure, HT16K33 LED Matrix Driver, PCA9685 16-channel PWM driver default address,BME280 Temp/Barometric/Humidity, BMP280 Temp/Barometric, TCA9548 1-to-8 I2C Multiplexer",
"0x77 PCA9685 16-channel PWM driver default address, TCA9548 1-to-8 I2C Multiplexer, HT16K33 LED Matrix Driver, IS31FL3731 144-LED Audio Modulated Matrix LED Driver (CharliePlex), BME280/BMP280 Temp/Barometric, MS5607/MS5611 Barometric Pressure, BMP180 Temp/Barometric, BMP085 Temp/Barometric, BMA180 Accelerometer, BME680/BME688 Low power gas, pressure, temperature & humidity sensor",
"0x78 PCA9685 16-channel PWM driver default address",
"0x79 PCA9685 16-channel PWM driver default address",
"0x7a PCA9685 16-channel PWM driver default address",
"0x7b PCA9685 16-channel PWM driver default address",
"0x7c PCA9685 16-channel PWM driver default address",
"0x7d PCA9685 16-channel PWM driver default address",
"0x7e PCA9685 16-channel PWM driver default address",
"0x7f PCA9685 16-channel PWM driver default address"
};

//----------

```
void setup()
{
  Serial.begin(115200);
  delay(500);
  pinMode(sda,OUTPUT);
  pinMode(scl,OUTPUT);
  Serial.println("\nI2C ADDRESS Scanner :  סורק כתובות ");
}

//-------------------------------------------------
void loop()
{
  Serial.println("Scanning  ...  סורק");
  for(address = 1,error=1; address < 128 ; address++ )
    {
      if(debug)
      {
        Serial.print("\nChecking device in address :  0x");
        Serial.print(address,HEX);
      }
      start();  //  שליחת start לרכיב
      sendByte(address<<1); // שליחת הכתובת לרכיב
      ack();  // המתנה לאישור מהרכיב
      if(error==0)  //  האם הרכיב החזיר אישור ?
      {
        Serial.println("   ------------- FOUND I2C Device   !!!   התגלה רכיב  ");
        Serial.print("Address is : 0x");
        Serial.println(address,HEX);
        Serial.print("Device is : \n");
        Serial.println(deviceAddress[address]);
        numberOfDevicesFound++;
        error=1;
        stop();
      }
```

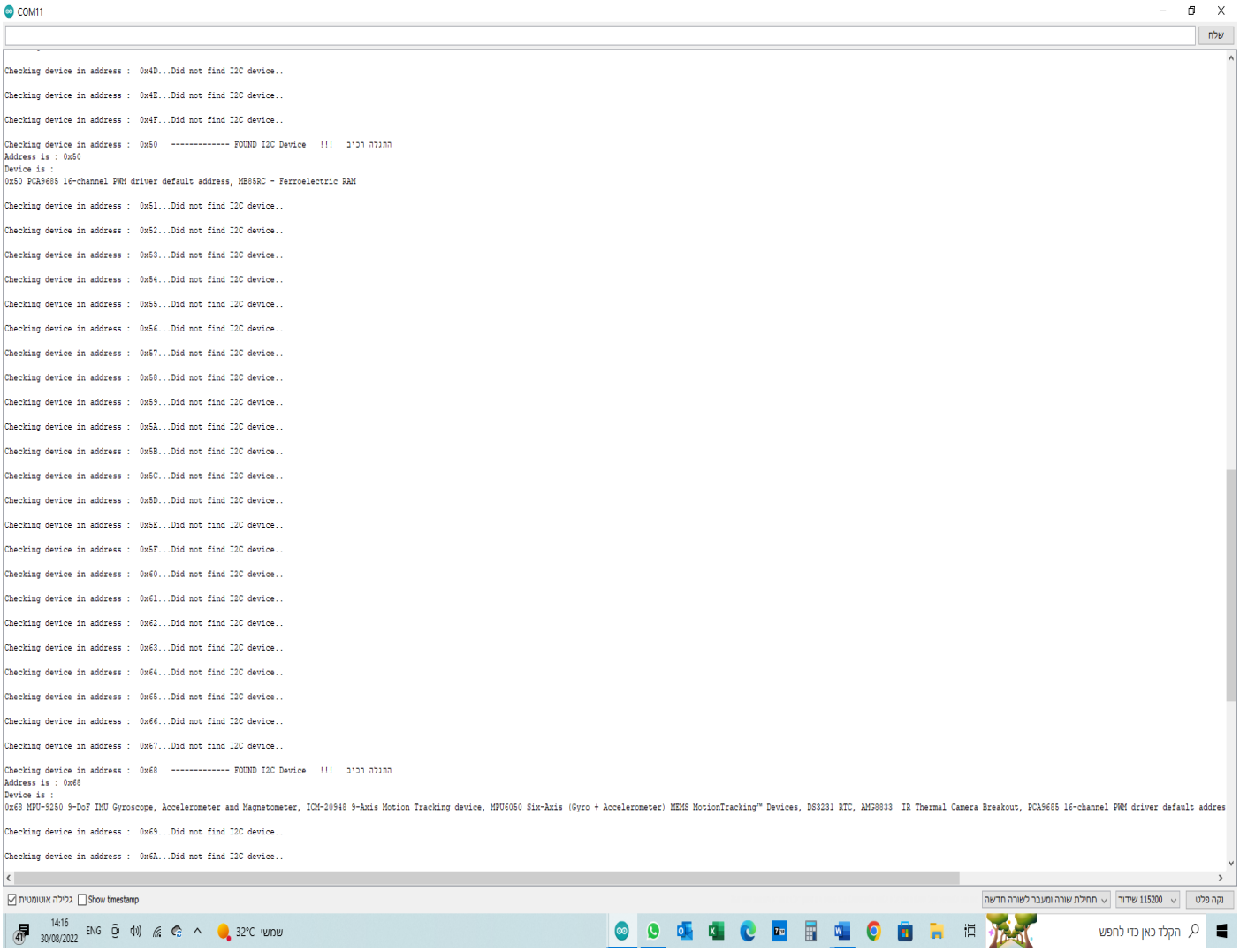
```
    else if(debug)
      Serial.println("...Did not find I2C device..");
 }
  Serial.println("\nProgram Ended  ----  התוכנית הסתיימה");
  Serial.print("Number of Addresses found : ");
  Serial.print(numberOfDevicesFound);
  Serial.println(" : כמות הכתובות שנמצאו ");
  while(1);
}

//--------- פונקציה השולחת 8 ביטים אחד אחרי השני בתקשורת I2C-------
//  השליחה היא מביט ה MSB אל ה LSB
void sendByte(byte data8)
{
  pinMode(sda,OUTPUT);
  for(unsigned char i=0;i<=7;i++)
   {
   digitalWrite(sda,data8/128);
    data8=data8<<1;
    digitalWrite(scl,1);
    delayMicroseconds(10);
    digitalWrite(scl,0);
  }
}

//---------------------------------------------
//  שינוי במצב קו הנתון מגבוה לנמוך כאשר השעון נמצא בגבוה
void start()
{
   digitalWrite(scl,1);
   digitalWrite(sda,1);
   digitalWrite(sda,0);
   digitalWrite(scl,0);
}
```

```
//----------------------------------------------
  // קו הנתון יהיה יציב בנמוך בזמן שקו השעון עולה לגבוה
void ack()
{
  unsigned long current,next;
  pinMode(sda,INPUT);// העברת קו הנתון לקלט כדי שהצד השני יוכל להוריד את הקו ל 0
  digitalWrite(scl,1);
// בלולאה נותנים עד 150 מילי שניות כדי לקבל אישור וברגע שיש 0 מסיימים את הלולאה
  for(next=current=millis();digitalRead(sda) && next-current<150;next=millis()) ;
  if(next-current<150)
    error=0;
  digitalWrite(scl,0);
  delay(1);
  pinMode(sda,OUTPUT);
}
//----------------------------------------------
// שינוי במצב קו הנתון מנמוך לגבוה כאשר השעון במצב גבוה
void stop()
{
  pinMode(sda,OUTPUT);
  digitalWrite(sda, 0);
  digitalWrite(scl, 1);
  digitalWrite(sda, 1);
}
```

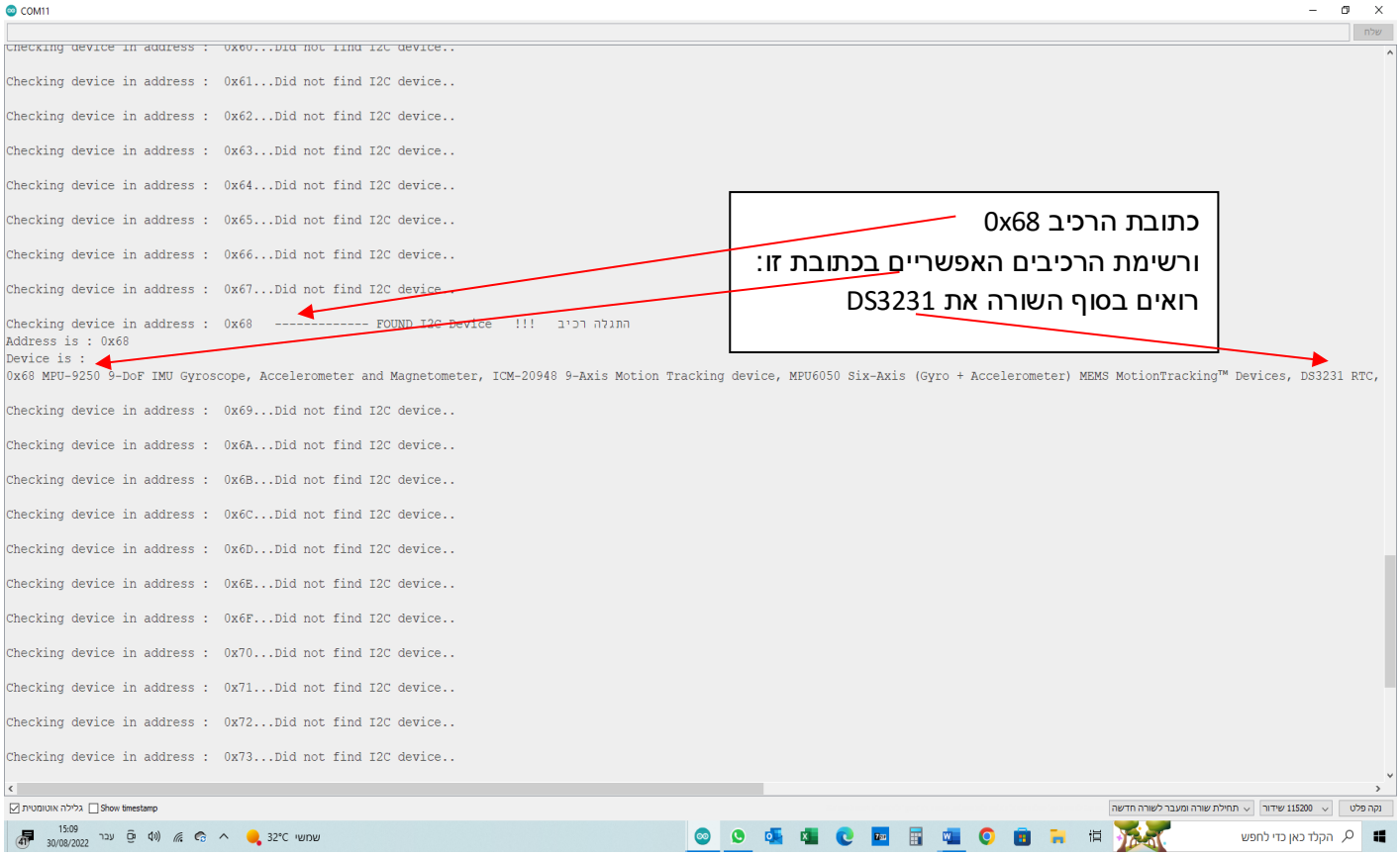
כאשר נשים בתוכנית בביט debug=1 ההדפסה שנקבל היא:



איור 12 : ההדפסה המתקבלת במוניטור הטורי עבור debug=1 ורכיב DS3231 .

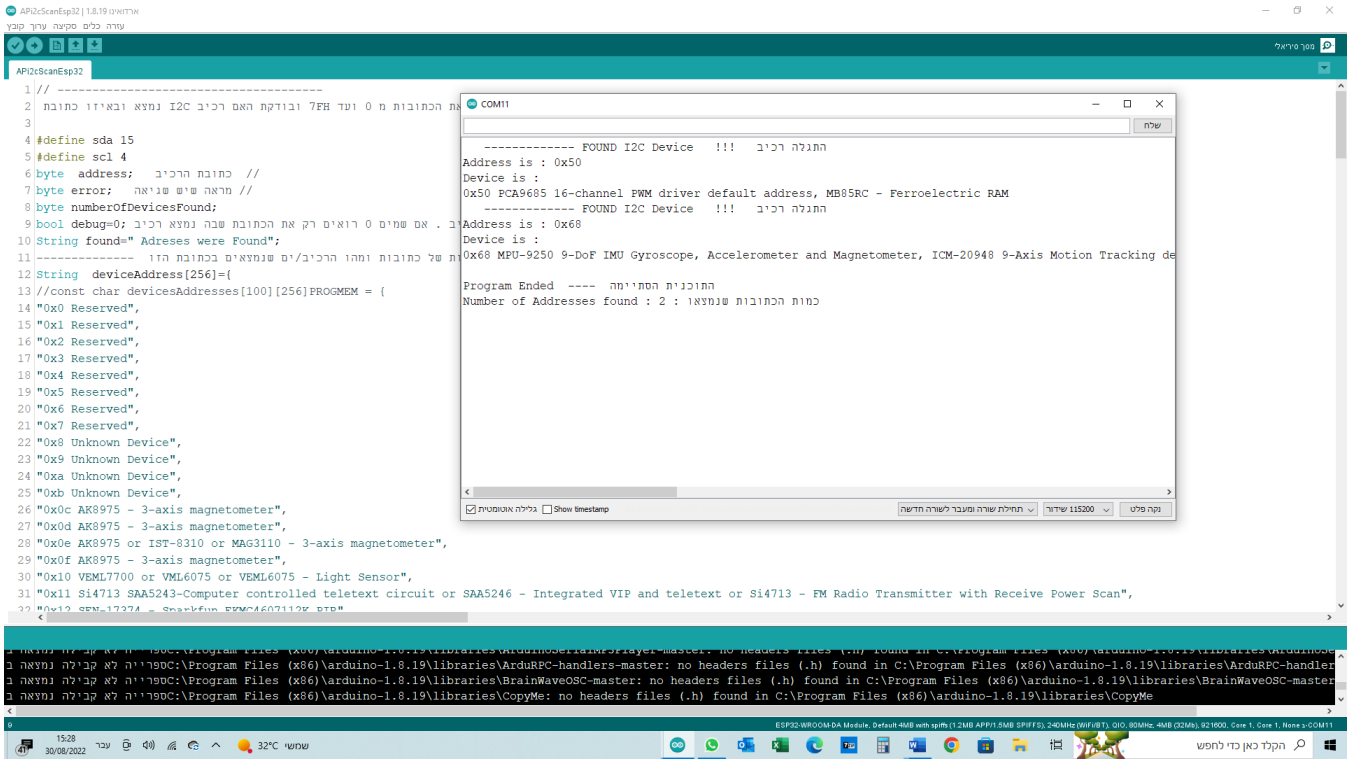
רואים במסך הטורי שהרכיב התגלה גם בכתובת 0x50 וגם בכתובת 0x68 . ציפינו לכתובת 0x68 . לא ברור למנה קיבלנו את כתובת 0x50 . קיבלנו כתובת זו גם בניסוי הקודם ושווה לחקור בעתיד מדוע הרכיב יושב על 2 כתובות (בעייה במודול ??) וזה יהיה בעבודה הבאה.....

אם נגדיל את המסך כדי שנראה מה מקבלים :



איור 13 : הגדלת מסך האיור הקודם.

אם נשים בביט debug=0 נקבל במסך המוניטור הטורי



איור 14 : המסך הטורי עבור debug=0 .

ה. ביבליוגרפיה:

1. האתר של Randomnerd בקישור:

https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/

2.

https://dl.espressif.com/dl/package_esp32_index.json

3. איור ההדקים של הכרטיס ESP32 DOIT DEV KIT V1

ESP32-DOIT-DEV-KIT-v1-pinout-mischianti.png (1685×825)