הספרייה TinyGPS++

ההסברים בעמודים הבאים מתבססים על המאמר שנכתב על ידי Mikal Hart באתר

http://arduiniana.org/libraries/tinygpsplus/

כדי להבין את הכתוב כאן יש להבין מהו GPS , מהם משפטי NMEA וכו' .

1. כללי ודוגמאות פשוטות

זוהי ספריה של הארדואינו המנתחת נתוני NMEA המתקבלים ממודולים של GPS .

בספרייה יש מתודות (כמו פונקציות או שגרות בשפת C) לחילוץ של מיקום, זמן , תאריך, גובה, מהירות וכיוון מרכיבי GPS.

את התכנה ניתן להוריד ב https://github.com/mikalhart/TinyGPSPlus/releases

ולצרף אותה אל ספריית libraries של הארדואינו .

הספרייה TinyGPS++ היא היורשת של הספרייה TinyGPS . היא גדולה יותר ומשתמשת באובייקטים ובמאפיינים שימושיים יותר.

נניח שיש לנו ארדואינו המחובר לרכיב GPS ורוצים להציג את הגובה. לשם כך נרשום את השורות הבאות :

```
1  #include "TinyGPS++.h"
2  TinyGPSPlus gps;
```

בשורה הראשונה אמרנו לקומפיילר להכליל את הספרייה TinyGPS++ .

בשורה השנייה ייצרנו אובייקט הנקרא gps ושייך למחלקה TinyGPS++ .

2השורות הבאות בודקות האם יש נתון זמין (שורה 3) ואם כן קוראים אותו (קוראים אותו .

```
1  while (ss.available() > 0)
2    gps.encode(ss.read());
```

ב 2 השורות הבאות נבדוק האם עודכן נתון הגובה ואם כן נדפיס את הגובה במטרים למסך המוניטור הטורי.

```
if (gps.altitude.isUpdated())
  Serial.println(gps.altitude.meters());
```

דוגמה נוספת להדפסת קו הרוחב, קו האורך והגובה נמצא ב 3 השורות הבאות :

```
Serial.print("LAT=");     Serial.println(gps.location.lat(), 6);
Serial.print("LONG=");    Serial.println(gps.location.lng(), 6);
Serial.print("ALT=");     Serial.println(gps.altitude.meters());
```

המספר 6 בסיום כל אחד מ 2 המשפטים שלמעלה אומר את מספר השדה של משפט ה NMEA שממנו רוצים לקבל את הנתונים .

כדי ש ++TinyGPS יעבוד יש לנתב את התווים אליו ממודול ה GPS בעזרת המתודה **encode()** . לדוגמה : אם חיברנו את מודול ה GPS אל הדקים 4 (קליטה) ו 3 (שידור) נרשום :

```
SoftwareSerial ss(4, 3);  // יצירת אובייקט לתקשורת טורית בהדקים 4 ו 3
void loop()
{
  while (ss.available() > 0)    // האם קלטנו נתון בתקשורת הטורית ?
    gps.encode(ss.read());      // משיכת הנתון שנקלט
  ...
```

אחרי שהאובייקט קיבל את כל הנתונים ניתן לבדוק האם כל שדות הנתונים שלו קיבלו את הנתונים בעזרת המשפט if (gps.location.isUpdated()) שבשורות הבאות ואם כן מדפיסים את קו הרוחב והאורך בעזרת הערכים החוזרים מהפונקציות gps.location.lat(), 6 ו gps.location.lng(), 6 בהתאמה.

```
if (gps.location.isUpdated())
{
  Serial.print("LAT="); Serial.print(gps.location.lat(), 6);
  Serial.print("LNG="); Serial.println(gps.location.lng(), 6);
}
```

האובייקט הראשי ב ++TinyGPS מכיל אובייקטים משניים של :

מיקום - location , תאריך – date , זמן - time , מהירות – speed , כיוון – course , גובה – altitude , כמות הלוויינים הנראים , - satellites , הפחתה (הורדה) של דיוק אופקי – hdop (– horizontal diminution of precision)

נרשום את כל המתודות של הספרייה (מדפיסים את הערכים למסך המוניטור הטורי) – ליד המאפיינים החשובים רשמנו את המאפיין בעברית:

```
Serial.println(gps.location.lat(), 6); // Latitude in degrees (double) – קו הרוחב במעלות

Serial.println(gps.location.lng(), 6); // Longitude in degrees (double) – קו אורך במעלות

Serial.print(gps.location.rawLat().negative ? "-" : "+") // הדפסת – או + של קו הרוחב

Serial.println(gps.location.rawLat().deg); // Raw latitude in whole degrees

Serial.println(gps.location.rawLat().billionths);// ... and billionths (u16/u32)

Serial.print(gps.location.rawLng().negative ? "-" : "+");

Serial.println(gps.location.rawLng().deg); // Raw longitude in whole degrees

Serial.println(gps.location.rawLng().billionths);// ... and billionths (u16/u32)

Serial.println(gps.date.value()); // Raw date in DDMMYY format (u32) - תאריך

Serial.println(gps.date.year()); // Year (2000+) (u16) - השנה

Serial.println(gps.date.month()); // Month (1-12) (u8) - החודש

Serial.println(gps.date.day()); // Day (1-31) (u8)

Serial.println(gps.time.value()); // Raw time in HHMMSSCC format (u32)

Serial.println(gps.time.hour()); // Hour (0-23) (u8) - השעה

Serial.println(gps.time.minute()); // Minute (0-59) (u8) - הדקות

Serial.println(gps.time.second()); // Second (0-59) (u8) - השניות

Serial.println(gps.time.centisecond()); // 100ths of a second (0-99) (u8) – מאיות שנייה

Serial.println(gps.speed.value()); // Raw speed in 100ths of a knot (i32)– יחידות- קשר ימי

Serial.println(gps.speed.knots()); // Speed in knots (double)

Serial.println(gps.speed.mph()); // Speed in miles per hour (double) – מהירות במייל/שעה

Serial.println(gps.speed.mps()); //Speed in meters per second (double)- מהירות במטר לשנייה

Serial.println(gps.speed.kmph()); // Speed in kilometers per hour (double)- בק"מ לשעה

Serial.println(gps.course.value()); // Raw course in 100ths of a degree (i32)

Serial.println(gps.course.deg()); // Course in degrees (double)
```

Serial.println(gps.altitude.value()); // Raw altitude in centimeters (i32) – גובה בס"מ

Serial.println(gps.altitude.meters()); // Altitude in meters (double) – גובה במטר

Serial.println(gps.altitude.miles()); // Altitude in miles (double) – גובה במייל

Serial.println(gps.altitude.kilometers()); // Altitude in kilometers (double) – גובה בק"מ

Serial.println(gps.altitude.feet()); // Altitude in feet (double) – גובה ברגל

Serial.println(gps.satellites.value()); // Number of satellites in use (u32) – כמות הלווויינים בשימוש

Serial.println(gps.hdop.value()); // Horizontal Dim. of Precision (100ths-i32) – הפחתת דיוק אופקי

2. עדכון ותקפות הנתונים

ניתן לבחון את ערכי האובייקט בכל זמן רצוי אבל כדאי לבדוק שאכן הנתונים תקפים לזמן הבדיקה.
המתודה (הפונקציה) ( )isValid אומרת האם האובייקט מכיל נתונים תקפים וזה בטוח למשוך אותם.

באופן דומה גם הפונקציה ( )isUpdated מציינת האם ערכי האובייקט עודכנו (לא חייב ששונו) מאז הקריאה האחרונה של הנתונים .

אם רוצים לדעת כמה זמן לא עודכנו הנתונים משתמשים בפונקציה ( )age המחזירה את כמות המילי שניות מאז העדכון האחרון. אם הערך המצוין גדול מ 1500 זה יכול לסמן בעיה כמו איבוד מיקום.

3. ניפוי – debugging

כאשר התכנית שרשמנו לא מצליחה לקבל נתונים זה בדרך כלל בגלל שהאובייקט איננו מקבל נתוני NMEA מלאים ואולי בכלל לא מקבלים נתונים. קיימות מספר מתודות העוזרות לאתר את הבעיה.

המתודות הן :

הפונקציה **( ) CharsProcessed** אומרת לנו כמה תווים נקלטו על ידי האובייקט .

הפונקציה **()sentencesWithFix** - מציינת את כמות משפטי ה $GPRMC או $GPGGA שיש להם fix (איכון , נעילה).

הפונקציה **failedChecksum()** שמציינת את מספר המשפטים מכל הסוגים שנכשלו בבדיקת checksum .

הפונקציה **passedChecksum()** שמציינת את מספר המשפטים מכל הסוגים שעברו את בדיקת checksum .

אם הפונקציה **CharsProcessed ( )** מחזירה 0 כנראה שיש בעיית חומרה . בדרך כלל כדאי לרשום בתחילת התכנית את השורות הבאות שמראות שאם לא קלטנו במשך 5 שניות מהרצת התכנית נתונים אז יש בעיה :

```
if (millis() > 5000 && gps.charsProcessed() < 10) // אם עברו 5 שניות וקלטנו פחות מ 5 תווים
{
  Serial.println("ERROR: not getting any GPS data!"); // הדפסה: שגיאה לא קולט נתונים
  // dump the stream to Serial
  Serial.println("GPS stream dump:");
  while (1) // לולאה אין סופית
    if (ss.available() > 0)    //    אם נקלט משהו בתקשורת ?
      Serial.write(ss.read());  //  הדפסת התו שנקלט
}
```

עוד תקלה נפוצה היא כאשר המשפטים המשודרים אל ה TinyGPS++ הם לא בשלמות. זה קורה בדרך כלל כאשר מקבלים את הנתונים לאט מאד או לפעמים רחוקות כך שחלק מהמשפטים אובד. נשתמש בפונקציה **failedChecksum()** שתראה לנו כמה משפטים נכשלו בבדיקת checksum . כזכור כל משפט NMEA מסתיים עם שדה מספרי המייצג סיכום מתמטי של כל התווים במשפט. זאת כדי להבטיח את שלמות הנתונים. אם המספר איננו תואם את הסיכום המעשי (ייתכן שמספר תווים אבדו) ואז tinyGPS++ מתעלם מכל המשפט ומגדיל מונה של כמות נכשלים checksum .

כדי לבדוק זאת נרשום את קטע התכנית הבא:

```
Serial.print("Sentences that failed checksum=");
Serial.println(gps.failedChecksum());
```

נבדוק האם הייתה גלישה בתכנת התקשורת הטורית SoftwareSerial בעזרת המשפטים :

```
// Testing overflow in SoftwareSerial is sometimes useful too.
Serial.print("Soft Serial device overflowed? ");
Serial.println(ss.overflow() ? "YES!" : "No");
```

אם המספר של כמות נכשלים של ה checksum הולך וגדל יש בעיה.

4. חילוץ הנתון הרצוי ממשפט NMEA

ניתן לחלץ ממשפט NMEA את הנתון מהשדה הרצוי . הרעיון הוא לומר ל ++TinyGPS את שם המשפט ואת מספר השדה הרצוי. לדוגמה (בהנחה שיצרנו אובייקט בשם magneticVariation):

```
TinyGPSCustom magneticVariation(gps, "GPRMC", 10);
```

אומרים שרוצים את השדה העשירי במשפטי $GPRMC ואז אפשר לרשום את המשפטים הבאים כדי לקבל הדפסה :

```
if (magneticVariation.isUpdated())  // האם האובייקט קיבל נתון
{
  Serial.print("Magnetic variation is ");
  Serial.println(magneticVariation.value());
}
```

5. ביסוס fix (איכון , נעילה)

אובייקטים של ++TinyGPS תלויים בתכנית שאנחנו רושמים כדי להזרים נתונים של NMEA . כדי להבטיח שהנתונים רלוונטיים יש לעשות 3 דברים :

א. להמשיך זרימת נתוני MNEA אל האובייקט עם המתודה ( )encode .

ב. משפט ה NMEA חייב לעבור את מבחן ה checksum .

ג. משפטי ה NMEA צריכים לתת דיווח עצמי לאובייקטים של הספרייה שהם תקפים, כלומר אם משפט $GPRMC מדווח על תקפות של "V" (void) במקום "A" (active) או אם משפט $GPGGA מדווח fix (נעילה) של "0" (כלומר, אין fix), אז מתעלמים מהמיקום והגובה (אפילו שנתוני הזמן בסדר).

הערה : יכול לקחת מספר דקות עד שמתבסס fix (איכון, נעילה) במיוחד אם נסענו רחוק או עבר זמן רב מאז השימוש האחרון במודול.

6. מרחק וכיוון

אם לאפליקציה שלנו יש יישום לנקודת ציון של היעד , ניתן לחשב את המרחק והכיוון אל נקודת הציון . יש 2 מתודות לביצוע הדבר וישנה מתודה שלישית **cardinal( )** להצגת הכיוון בצורה ידידותית , שניתנת לקריאה כמצפן כיוון.

קטע התכנית הבא מוצא מרחק וכיוון:

```
const double EIFFEL_TOWER_LAT = 48.85826;  // קואורדינטת הרוחב של מגדל אייפל
const double EIFFEL_TOWER_LNG = 2.294516; // קואורדינטת האורך של מגדל אייפל
// נזמן מתודה שמוצאת את המרחק בין הקואורדינטות הנוכחיות ואלו של מגדל אייפל
double distanceKm =
  TinyGPSPlus.distanceBetween(        gps.location.lat(),
    gps.location.lng(),
    EIFFEL_TOWER_LAT,
    EIFFEL_TOWER_LNG) / 1000.0;
// נזמן מתודה שתמצא את הכיוון מהקואורדינטות הנוכחיות אל מגדל אייפל
double courseTo =
  TinyGPSPlus.courseTo(
    gps.location.lat(),
    gps.location.lng(),
    EIFFEL_TOWER_LAT,
    EIFFEL_TOWER_LNG);
// הדפסות למסך
Serial.print("Distance (km) to Eiffel Tower: ");
Serial.println(distanceKm);
Serial.print("Course to Eiffel Tower: ");
Serial.println(courseTo);
Serial.print("Human directions: ");
Serial.println(TinyGPSPlus.cardinal(courseTo));
```

כדי לבדוק האם האובייקטים של TinyGPS++ מכילים נתוני fix תקפים , יש להעביר את הכתובת של משתנה מטיפוס unsigned long כפרמטר אל המתודה fix_age . אם הערך החוזר הוא TinyGPS::GPS_INVALID_AGE יודעים שהאובייקט לא קיבל אף פעם fix תקף. (סימן השיוך :: מראה שמדובר בפונקציית חבר - member function - השייכת למחלקה tinyGPS). אם חוזר ערך הוא מציין את מספר מיליוניות השנייה מאז בוצע fix תקף. אם מבצעים עדכון נתונים קבוע של האובייקט הוא לא יעלה על 1000 . אם fix_age הולך וגדל זה מראה שהייה לנו fix אבל איבדנו אותו. קטע התכנית הבא יכול לבדוק אם יש fix (איכון).

```
float flat, flon; // הגדרת משתנים
unsigned long fix_age;  // משתנה שיחזור אליו + - לרוחב ואורך במעלות
```

```
gps.f_get_position(&flat, &flon, &fix_age); // זימון המתודה
if (fix_age == TinyGPS::GPS_INVALID_AGE)  // אם חוזר ערך לא תקף
  Serial.println("No fix detected");
else if (fix_age > 5000)  // 5000 אם הזמן שהיה איכון אחרון גדול מ
  Serial.println("Warning: possible stale data!"); //  "אפשרות נתונים ישנים" : הדפסה
else
 Serial.println("Data is current."); //    "הנתונים עכשוויים " : הדפסה
```

7. בדיקת הגרסה של הספרייה

ניתן לדעת את הגרסה של ספריית ה ++TinyGPS בעזרת קריאה לפונקציה החבר הסטטית libraryVersion() .

שורת ההדפסה המזמנת את הפונקציה ומדפיסה למוניטור הטורי :

```
Serial.println(TinyGPSPlus::libraryVersion());
```

8. תכניות דוגמה המסופקות עם הספרייה

בספרייה מספר תכניות דוגמה מהפשוטה אל המורכבת.
מתחילים בתכנית BasicExample מדגימה עקרונות בסיסיים ללא צורך ברכיב GPS. עוברים לתכנית FullExample ו KitchenSink ולבסוף יש תכניות שהמשתמש יכול לחלץ את השדות המעניינים אותו.