Portable Stand Alone Grid Square Display
Bill Pence KI4US

Ever thought about needing a portable grid square display for a rove or portable operation?
Looking for stand alone, low cost, flexible and easy to assemble?

When Brian, NX9O was planning a long EME rove (http://www.rfacres.com/EME/), he posed a query to the Fourlanders Slack channel about a stand alone grid display. Several answers of "Griduino" were offered. Of course, the https://griduino.com/ is really pretty, and comes with somewhat steep price but the web page shows the kit is constantly out of stock.

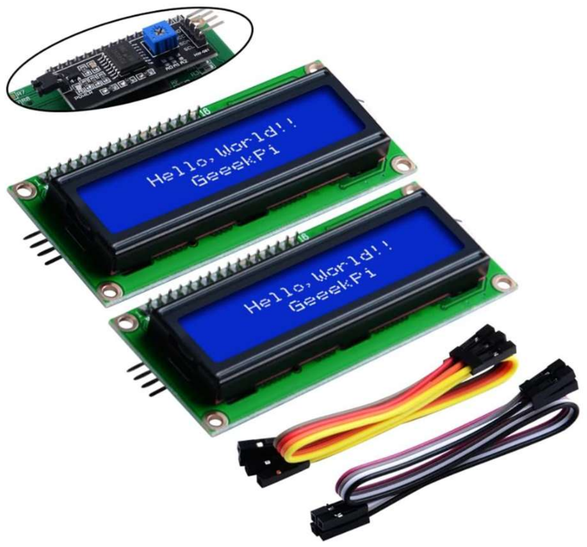
Later, I did find the BOM (Bill of Materials) for Griduino, and it seemed all of the parts could be had across several vendors.

Mobile phone apps for iPhone and Android exist but monopolize the phone display.

However, in the interim, Brian and I had explored alternatives. Several searches turned up a function for Arduino by Glen Popiel (KW5GP) which seemed promising.

So….
Brian and I created a really simple and affordable grid square display unit with easy to obtain Arduino modules.
This project hardware requires an Arduino of most any flavor, a GPS module, and an I2C LCD character display. It would be advisable to use an Arduino with a built in 5v regulator so that direct 12V DC power can be provided.
(that is really everything… besides of course the Arduino Integrated Development Environment (IDE) from https://www.arduino.cc/ on a computer.)

Here is an example of the LCD and GPS modules and a low cost Arduino Uno clone.
https://www.amazon.com/GeeekPi-Character-Backlight-Raspberry-Electrical/dp/B07S7PJYM6
https://www.amazon.com/gp/product/B01D1D0F5M
https://www.amazon.com/ELEGOO-Board-ATmega328P-ATMEGA16U2-Compliant/dp/B01EWOE0UU/



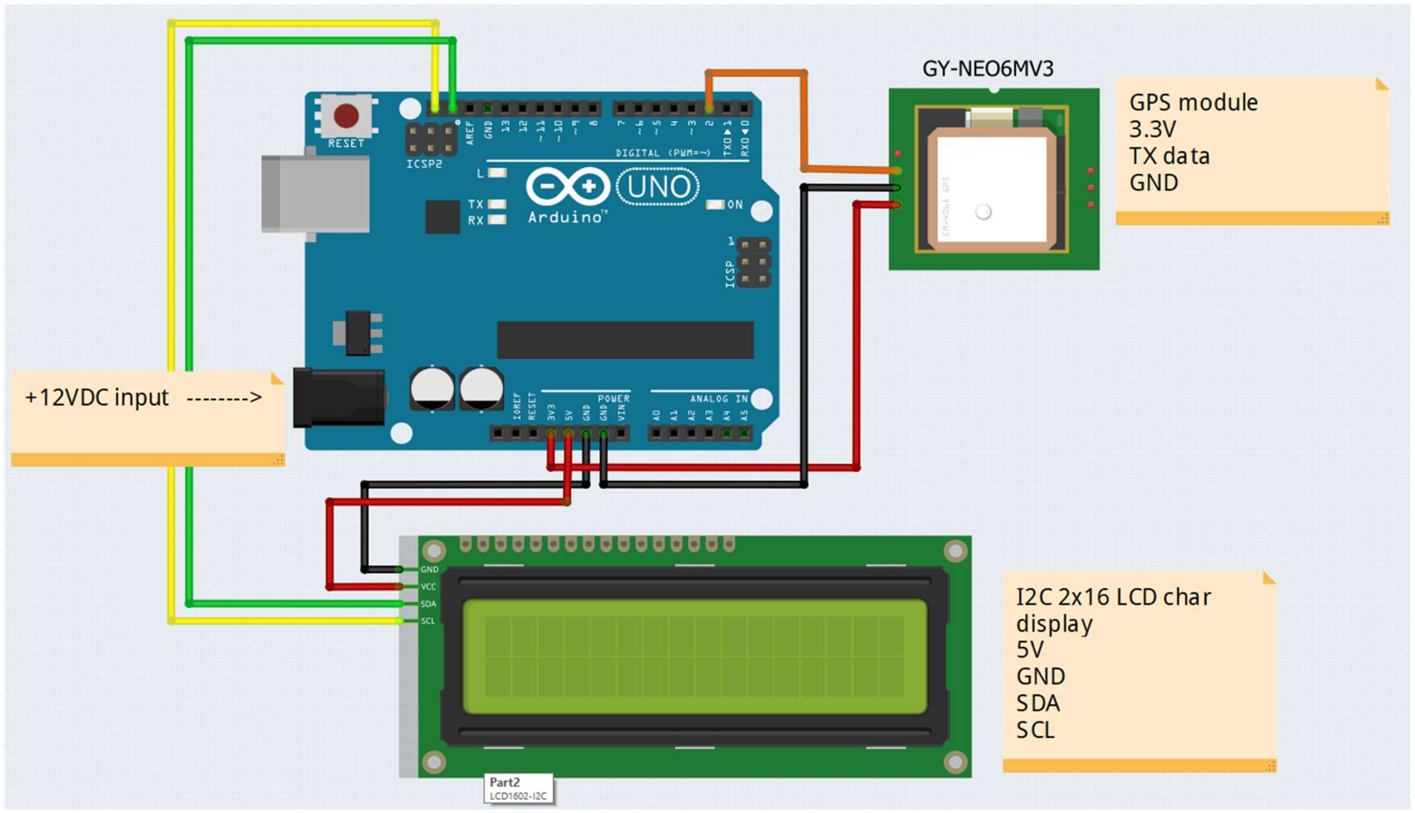
the github repository is located at https://github.com/bpguy/gps_grid_sq_lcd and contains the arduino sketch, this document, and the libraries needed (in case they are not easy to find.)

The github should be openly accessible.

Wiring is relatively simple. The GPS module requires only 3 wires: 3.3 VCC, GND, and TX data. (Since no data is sent to the GPS module, only the GPS serial TX line is required.) Arduino jumpers are usually included with most kits, but are easily found on Amazon.com or ebay.com.

The LCD module requires 4 wires: 5V VCC, GND, SDA and SCL.

Use this handy diagram to attach the LCD and GPS module.



If you do not have much Arduino experience, there are MANY quick learning modules on the web. Adafruit, randomnerd, and circuit geeks have good tutorials
https://learn.adafruit.com/category/learn-arduino
https://www.circuitgeeks.com/. YouTube has Arduino learning as well.

Here are tutorials to provide more info on using i2C LCD and GPS modules.
https://www.circuitgeeks.com/arduino-i2c-lcd-tutorial/
https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/

One important note is that the I2C address for the LCD display may not be the same as others. (Brian's display was a different I2Caddress from the one I used to develop the sketch...) so the address will need to be discovered and may need updated. The LCD contrast pot on back of the LCD may need adjusting as well.

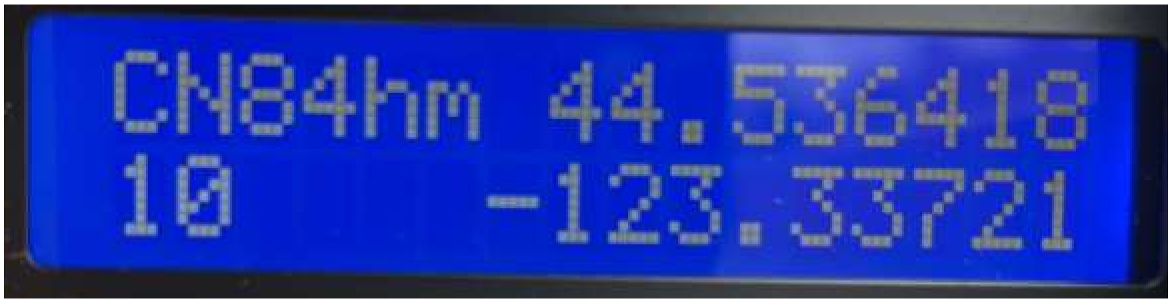
An I2C "scanner" sketch, like the one provided from adafruit, will be used to find the address.
https://learn.adafruit.com/scanning-i2c-addresses/arduino
Once the I2C address for the LCD display is obtained, open the grid square sketch and put that LCD address into the code at line 21. (I'd also suggest writing the address on the module as well for future reference.)

```
#define LCD_ADDRESS 0x27
```

Once the I2C address is confirmed and put into the sketch, download the arduino sketch to the Arduino Uno (or similar) and the place the hardware setup in a place to get GPS fixes. It can take many minutes for first fix…

In use, on power on, the GPS module will try to get a fix. Once this occurs, the Arduino will begin decoding and displaying the GPS info to the LCD display. The GPS info is also sent via the serial port on the Arduino (via USB if the Arduino has USB…) and can be displayed.

Here is an example of the LCD display showing the current 6 digit grid square, the latitude and longitude and the number of satellites in view.



```
CN84hm 44.536418
10    -123.33721
```

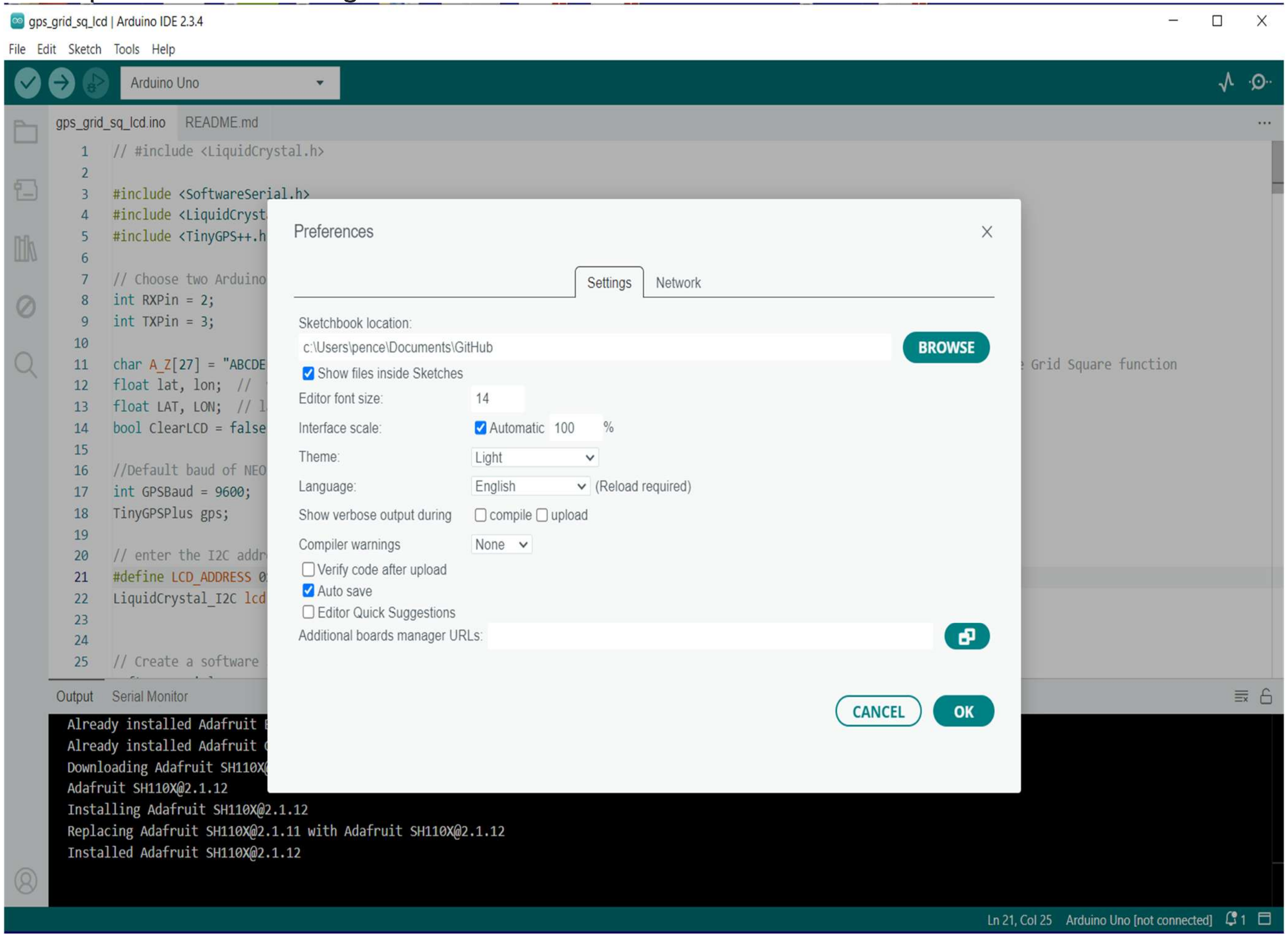
Bread boarding and mounting is flexible, including perhaps a cardboard case with tape.
Power can be as simple as providing 12v to the arduino if it includes an onboard 5v regulator. The GPS and LCD power requirements did not tax the Arduno Uno clone I used.

The arduino libraries needed are TinyGPSPlus and LiquidCrystal_I2C (see notes below)

The Arduino sketch is included at the end of this document. Please grab this from the git repository rather than typing this.

here is a brief description of setting up the Arduino IDE

You can see the sketch folder location in the IDE menu
file -> preferences settings.


gps_grid_sq_lcd | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Uno
gps_grid_sq_lcd.ino
README.md
Output
Serial Monitor
Ln 21, Col 25 Arduino Uno [not connected]

```
// #include <LiquidCrystal.h>

#include <SoftwareSerial.h>
#include <LiquidCryst
#include <TinyGPS++.h

// Choose two Arduino
int RXPin = 2;
int TXPin = 3;

char A_Z[27] = "ABCDE                                                   Grid Square function
float lat, lon;  //
float LAT, LON;  // l
bool ClearLCD = false

//Default baud of NEO
int GPSBaud = 9600;
TinyGPSPlus gps;

// enter the I2C addr
#define LCD_ADDRESS 0
LiquidCrystal_I2C lcd


// Create a software
```

Preferences

Settings | Network

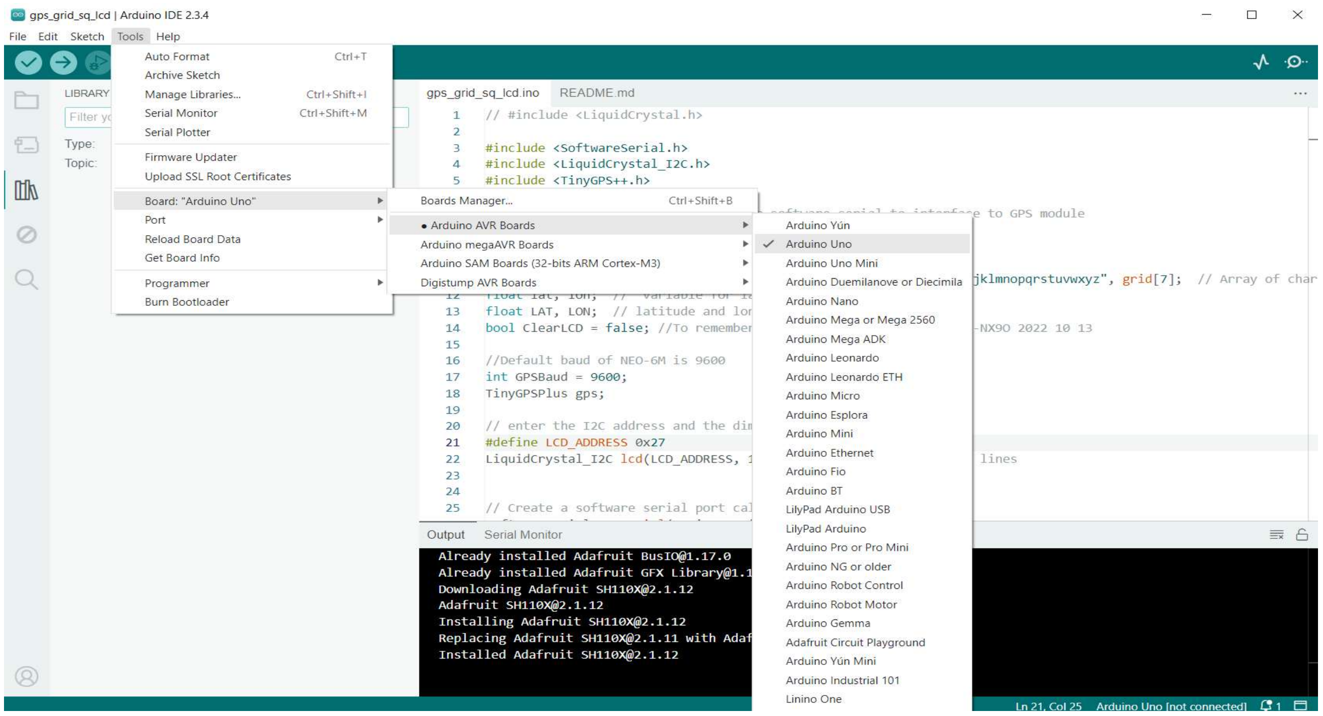
Sketchbook location: c:\Users\pence\Documents\GitHub BROWSE
[✓] Show files inside Sketches
Editor font size: 14
Interface scale: [✓] Automatic 100 %
Theme: Light
Language: English (Reload required)
Show verbose output during [ ] compile [ ] upload
Compiler warnings None
[ ] Verify code after upload
[✓] Auto save
[ ] Editor Quick Suggestions
Additional boards manager URLs:

CANCEL OK

```
Already installed Adafruit
Already installed Adafruit
Downloading Adafruit SH110X
Adafruit SH110X@2.1.12
Installing Adafruit SH110X@2.1.12
Replacing Adafruit SH110X@2.1.11 with Adafruit SH110X@2.1.12
Installed Adafruit SH110X@2.1.12
```

Make a folder in your Arduino sketchbook named gps_grid_sq_lcd and put the github files into this folder. The folder and arduino sketch (.ino file) must have the same name.

then open the sketch and select Arduino Uno (or the board you are using) from the board drop down.



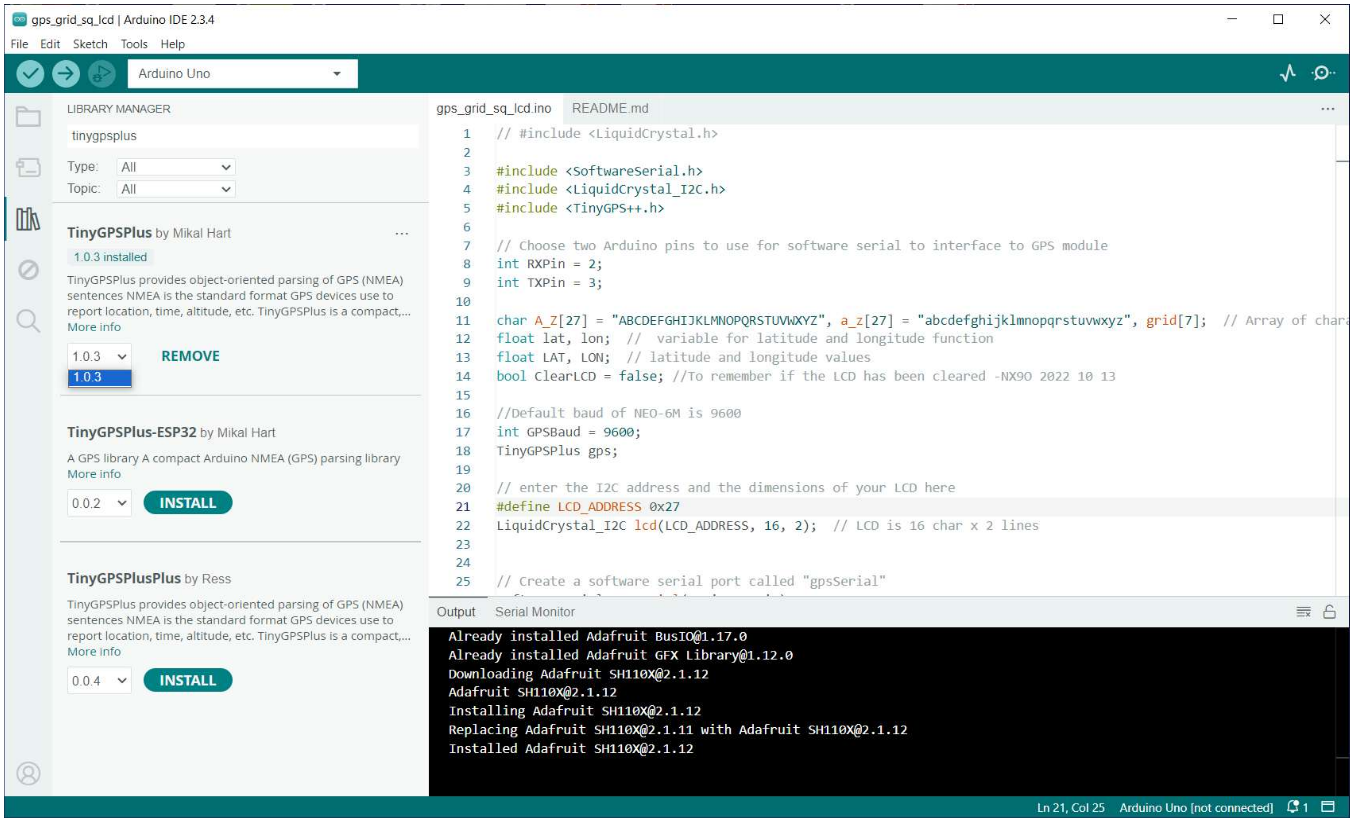
```
// #include <LiquidCrystal.h>

#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>
#include <TinyGPS++.h>
```

```
float LAT, LON;  // latitude and lon
bool ClearLCD = false; //To remember

//Default baud of NEO-6M is 9600
int GPSBaud = 9600;
TinyGPSPlus gps;

// enter the I2C address and the dim
#define LCD_ADDRESS 0x27
LiquidCrystal_I2C lcd(LCD_ADDRESS, 1


// Create a software serial port cal
```

```
Already installed Adafruit BusIO@1.17.0
Already installed Adafruit GFX Library@1.1
Downloading Adafruit SH110X@2.1.12
Adafruit SH110X@2.1.12
Installing Adafruit SH110X@2.1.12
Replacing Adafruit SH110X@2.1.11 with Adaf
Installed Adafruit SH110X@2.1.12
```

click the checkmark in the top left below “File” menu to try to compile. it should fail complaining about missing libraries.

in menu Tools -> Manage Libraries (or click the "books" on the left part of the window)
search for TinyGPSPlus and install the one shown in the list of libraries.



tinygpsplus

Type: All
Topic: All

TinyGPSPlus by Mikal Hart
1.0.3 installed
TinyGPSPlus provides object-oriented parsing of GPS (NMEA) sentences NMEA is the standard format GPS devices use to report location, time, altitude, etc. TinyGPSPlus is a compact,...
More info
1.0.3 REMOVE

TinyGPSPlus-ESP32 by Mikal Hart
A GPS library A compact Arduino NMEA (GPS) parsing library
More info
0.0.2 INSTALL

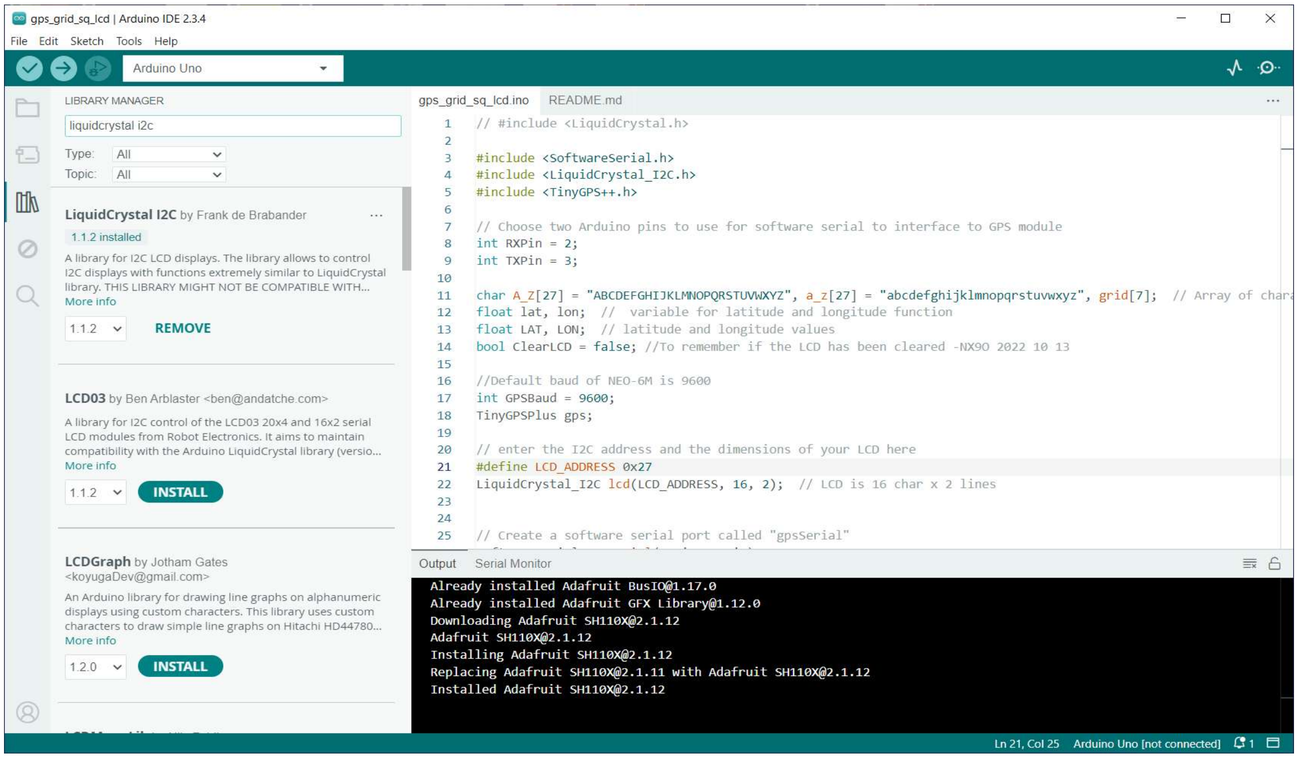
TinyGPSPlusPlus by Ress
TinyGPSPlus provides object-oriented parsing of GPS (NMEA) sentences NMEA is the standard format GPS devices use to report location, time, altitude, etc. TinyGPSPlus is a compact,...
More info
0.0.4 INSTALL

```
// #include <LiquidCrystal.h>

#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>
#include <TinyGPS++.h>

// Choose two Arduino pins to use for software serial to interface to GPS module
int RXPin = 2;
int TXPin = 3;

char A_Z[27] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ", a_z[27] = "abcdefghijklmnopqrstuvwxyz", grid[7];  // Array of chara
float lat, lon;  //  variable for latitude and longitude function
float LAT, LON;  // latitude and longitude values
bool ClearLCD = false; //To remember if the LCD has been cleared -NX9O 2022 10 13

//Default baud of NEO-6M is 9600
int GPSBaud = 9600;
TinyGPSPlus gps;

// enter the I2C address and the dimensions of your LCD here
#define LCD_ADDRESS 0x27
LiquidCrystal_I2C lcd(LCD_ADDRESS, 16, 2);  // LCD is 16 char x 2 lines


// Create a software serial port called "gpsSerial"
```

```
Already installed Adafruit BusIO@1.17.0
Already installed Adafruit GFX Library@1.12.0
Downloading Adafruit SH110X@2.1.12
Adafruit SH110X@2.1.12
Installing Adafruit SH110X@2.1.12
Replacing Adafruit SH110X@2.1.11 with Adafruit SH110X@2.1.12
Installed Adafruit SH110X@2.1.12
```

Ln 21, Col 25   Arduino Uno [not connected]

Then search LiquidCrystal I2C and install this one.


gps_grid_sq_lcd | Arduino IDE 2.3.4

LIBRARY MANAGER

liquidcrystal i2c

Type: All
Topic: All

LiquidCrystal I2C by Frank de Brabander
1.1.2 installed
A library for I2C LCD displays. The library allows to control I2C displays with functions extremely similar to LiquidCrystal library. THIS LIBRARY MIGHT NOT BE COMPATIBLE WITH...
More info
1.1.2 REMOVE

LCD03 by Ben Arblaster <ben@andatche.com>
A library for I2C control of the LCD03 20x4 and 16x2 serial LCD modules from Robot Electronics. It aims to maintain compatibility with the Arduino LiquidCrystal library (versio...
More info
1.1.2 INSTALL

LCDGraph by Jotham Gates <koyugaDev@gmail.com>
An Arduino library for drawing line graphs on alphanumeric displays using custom characters. This library uses custom characters to draw simple line graphs on Hitachi HD44780...
More info
1.2.0 INSTALL

gps_grid_sq_lcd.ino README.md

```
// #include <LiquidCrystal.h>

#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>
#include <TinyGPS++.h>

// Choose two Arduino pins to use for software serial to interface to GPS module
int RXPin = 2;
int TXPin = 3;

char A_Z[27] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ", a_z[27] = "abcdefghijklmnopqrstuvwxyz", grid[7];  // Array of chara
float lat, lon;  //  variable for latitude and longitude function
float LAT, LON;  // latitude and longitude values
bool ClearLCD = false; //To remember if the LCD has been cleared -NX9O 2022 10 13

//Default baud of NEO-6M is 9600
int GPSBaud = 9600;
TinyGPSPlus gps;

// enter the I2C address and the dimensions of your LCD here
#define LCD_ADDRESS 0x27
LiquidCrystal_I2C lcd(LCD_ADDRESS, 16, 2);  // LCD is 16 char x 2 lines


// Create a software serial port called "gpsSerial"
```

Output Serial Monitor

```
Already installed Adafruit BusIO@1.17.0
Already installed Adafruit GFX Library@1.12.0
Downloading Adafruit SH110X@2.1.12
Adafruit SH110X@2.1.12
Installing Adafruit SH110X@2.1.12
Replacing Adafruit SH110X@2.1.11 with Adafruit SH110X@2.1.12
Installed Adafruit SH110X@2.1.12
```

Ln 21, Col 25 Arduino Uno [not connected]

GPS grid Square arduino sketch

```
//#include <LiquidCrystal.h>

#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>
#include <TinyGPS++.h>

// Choose two Arduino pins to use for software serial to interface to
GPS module
int RXPin = 2;
int TXPin = 3;

char A_Z[27] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ", a_z[27] =
"abcdefghijklmnopqrstuvwxyz", grid[7];  // Array of characters for
the Grid Square function
float lat, lon;  //  variable for latitude and longitude function
float LAT, LON;  // latitude and longitude values
bool ClearLCD = false; //To remember if the LCD has been cleared -
NX9O 2022 10 13

//Default baud of NEO-6M is 9600
int GPSBaud = 9600;
TinyGPSPlus gps;

// enter the I2C address and the dimensions of your LCD here
#define LCD_ADDRESS 0x3F
LiquidCrystal_I2C lcd(LCD_ADDRESS, 16, 2);  // LCD is 16 char x 2
lines


// Create a software serial port called "gpsSerial"
SoftwareSerial gpsSerial(RXPin, TXPin);

void setup() {

  // Start the Arduino hardware serial port at 9600 baud
  Serial.begin(9600);

  // Start the software serial port at the GPS's default baud
  gpsSerial.begin(GPSBaud);

  //set up LCD
  lcd.init();
  lcd.clear();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("GPS GRID SQUARE"); //Added the word SQUARE - NX9O 2022
10 13
  Serial.println("GPS Grid Display");
  delay(5000);

  Serial.println(F("Sats HDOP  Latitude   Longitude   Fix  Date
Time     Date Alt    Course Speed Card  Chars Sentences Checksum"));
  Serial.println(F("          (deg)      (deg)       Age
Age  (m)    --- from GPS ----  RX    RX        Fail"));
  Serial.println(F("----------------------------------------------------
-------------------------------------------------------------------"));


}

void loop() {

  displayInfo();
}

void displayInfo() {

  printInt(gps.satellites.value(), gps.satellites.isValid(), 5);
  printFloat(gps.hdop.hdop(), gps.hdop.isValid(), 6, 1);
  printFloat(gps.location.lat(), gps.location.isValid(), 11, 6);
  printFloat(gps.location.lng(), gps.location.isValid(), 12, 6);
  printInt(gps.location.age(), gps.location.isValid(), 5);
  printDateTime(gps.date, gps.time);
  printFloat(gps.altitude.meters(), gps.altitude.isValid(), 7, 2);
  printFloat(gps.course.deg(), gps.course.isValid(), 7, 2);
  printFloat(gps.speed.kmph(), gps.speed.isValid(), 6, 2);
  printStr(gps.course.isValid() ?
TinyGPSPlus::cardinal(gps.course.deg()) : "*** ", 6);

  printInt(gps.charsProcessed(), true, 6);
  printInt(gps.sentencesWithFix(), true, 10);
  printInt(gps.failedChecksum(), true, 9);
  Serial.println();

  // write to the LCD as well



if (gps.location.isValid() )
  {
  lat = gps.location.lat();
  lon = gps.location.lng();
  LAT = lat*1000000;  // grid sq routine expects fix in slightly diff
format than tinygps++ library... (no decimal)
  LON = lon*1000000;

  GridSquare(LAT, LON);  // Calulate the Grid Square
  Serial.println(grid);

  // Blank the display if not already- NX9O 2022 10 13
  if (!ClearLCD)
  {
    lcd.clear();
    ClearLCD = true;
    //lcd.setCursor(0, 1);
    //lcd.print("Sat");
  }

  // Changed the order of LCD lines and removed the word "GRID" NX9O
2022 10 13
  //lcd.setCursor(0, 0);
  // lcd.print("GRID            ");
  lcd.setCursor(0, 0);
  lcd.print(grid);
  lcd.setCursor(0, 1);
  lcd.print(gps.satellites.value());

  lcd.setCursor(7, 0);
  lcd.print(lat,6);  // decimals was ,3 - NX9O 2022 10 13
  if (lon < 100)  // Move lat over 1 character if 100 degrees or
higher; later test for - and -100 - NX9O 2022 10 13
  {
    lcd.setCursor(5, 1);
    lcd.print(" ");
    lcd.setCursor(6, 1);
    lcd.print(lon,6);
  } else {
    lcd.setCursor(5, 1);
    lcd.print(lon,6);
  }

  }
  else
  {
  lcd.setCursor(0, 1);
  lcd.print("No Valid Fix    ");
  }

  smartDelay(1000);
  if (millis() > 5000 && gps.charsProcessed() < 10)
    Serial.println(F("No GPS data received: check wiring"));

  Serial.println();
  Serial.println();
}


// This custom version of delay() ensures that the gps object
// is being "fed".
static void smartDelay(unsigned long ms) {
  unsigned long start = millis();
  do {
    while (gpsSerial.available())
      gps.encode(gpsSerial.read());
  } while (millis() - start < ms);
}

static void printFloat(float val, bool valid, int len, int prec) {
  if (!valid) {
    while (len-- > 1)
      Serial.print('*');
    Serial.print(' ');
  } else {
    Serial.print(val, prec);
    int vi = abs((int)val);
    int flen = prec + (val < 0.0 ? 2 : 1);  // . and -
    flen += vi >= 1000 ? 4 : vi >= 100 ? 3
                           : vi >= 10  ? 2
                                       : 1;
    for (int i = flen; i < len; ++i)
      Serial.print(' ');
  }
  smartDelay(0);
}

static void printInt(unsigned long val, bool valid, int len) {
  char sz[32] = "*****************";
  if (valid)
    sprintf(sz, "%ld", val);
  sz[len] = 0;
  for (int i = strlen(sz); i < len; ++i)
    sz[i] = ' ';
  if (len > 0)
    sz[len - 1] = ' ';
  Serial.print(sz);
  smartDelay(0);
}
```

```
static void printDateTime(TinyGPSDate &d, TinyGPSTime &t) {
  if (!d.isValid()) {
    Serial.print(F("********** "));
  } else {
    char sz[32];
    sprintf(sz, "%02d/%02d/%02d ", d.month(), d.day(), d.year());
    Serial.print(sz);
  }

  if (!t.isValid()) {
    Serial.print(F("******** "));
  } else {
    char sz[32];
    sprintf(sz, "%02d:%02d:%02d ", t.hour(), t.minute(), t.second());
    Serial.print(sz);
  }

  printInt(d.age(), d.isValid(), 5);
  smartDelay(0);
}

static void printStr(const char *str, int len) {
  int slen = strlen(str);
  for (int i = 0; i < len; ++i)
    Serial.print(i < slen ? str[i] : ' ');
  smartDelay(0);
}

// this is the function I found from another project... it used a
diff GPS library so I had to change to fix format to accomodate...
// https://github.com/ke7uia/GridSquareDisplay

void GridSquare(float latitude,float longtitude)  // Calculate the
Grid Square from the latitude and longitude
{
  // Maidenhead Grid Square Calculation
  float lat_0,lat_1, long_0, long_1, lat_2, long_2, lat_3,
long_3,calc_long, calc_lat, calc_long_2, calc_lat_2, calc_long_3,
calc_lat_3;  // Set up the function variables
  lat_0 = latitude/1000000;
  long_0 = longtitude/1000000;

  int grid_long_1, grid_lat_1, grid_long_2, grid_lat_2, grid_long_3,
grid_lat_3;

  // Begin Calcs
  calc_long = (long_0 + 180);  // Calculate the first 2 characters of
the Grid Square
  calc_lat = (lat_0 + 90);
  long_1 = calc_long/20;
  lat_1 = (lat_0 + 90)/10;

  grid_lat_1 = int(lat_1);
  grid_long_1 = int(long_1);

  calc_long_2 = (long_0+180) - (grid_long_1 * 20);  // Calculate the
next 2 digits of the Grid Square
  long_2 = calc_long_2 / 2;
  lat_2 = (lat_0 + 90) - (grid_lat_1 * 10);
  grid_long_2 = int(long_2);
  grid_lat_2 = int(lat_2);

  calc_long_3 = calc_long_2 - (grid_long_2 * 2);  // Calculate the
last 2 characters of the Grid Square
  long_3 = calc_long_3 / .083333;
  grid_long_3 = int(long_3);

  lat_3 = (lat_2 - int(lat_2)) / .0416665;
  grid_lat_3 = int(lat_3);

  // Here's the first 2 characters of Grid Square - place into array
  grid[0] = A_Z[grid_long_1];
  grid[1] = A_Z[grid_lat_1];

  // The second set of the grid square
  grid[2] = (grid_long_2 + 48);  // ascii 0 0x30
  grid[3] = (grid_lat_2 + 48);

  // The final 2 characters
  grid[4] = a_z[grid_long_3];
  grid[5] = a_z[grid_lat_3];

  return;
```