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.

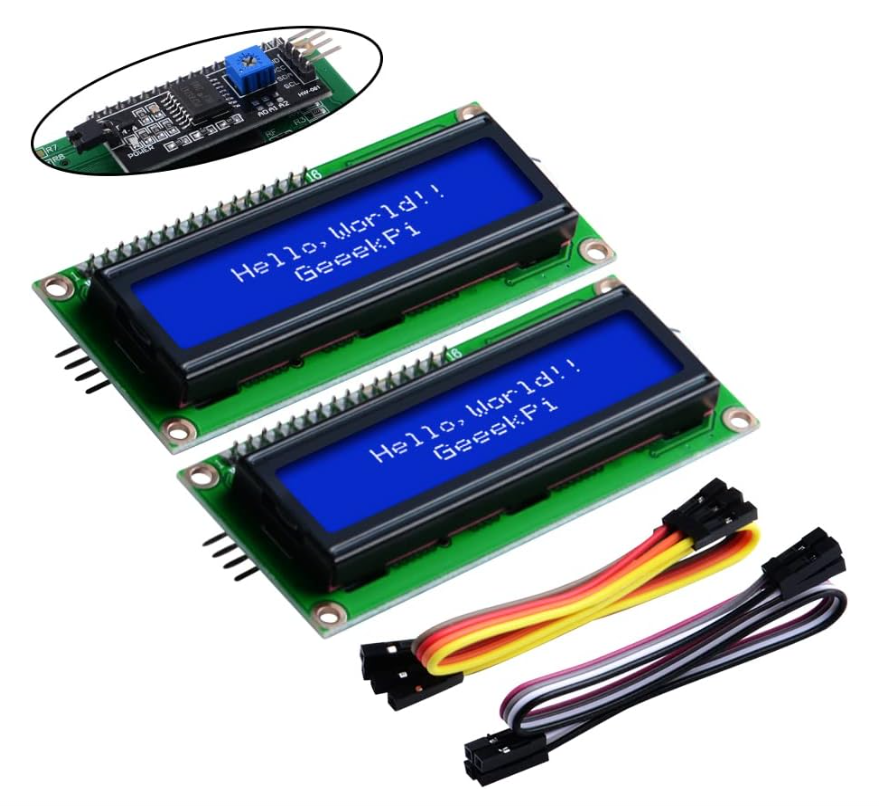
(that is really everything… besides of course the Arduino Integrated Development Environment (IDE) from <https://www.arduino.cc/> on a computer.)

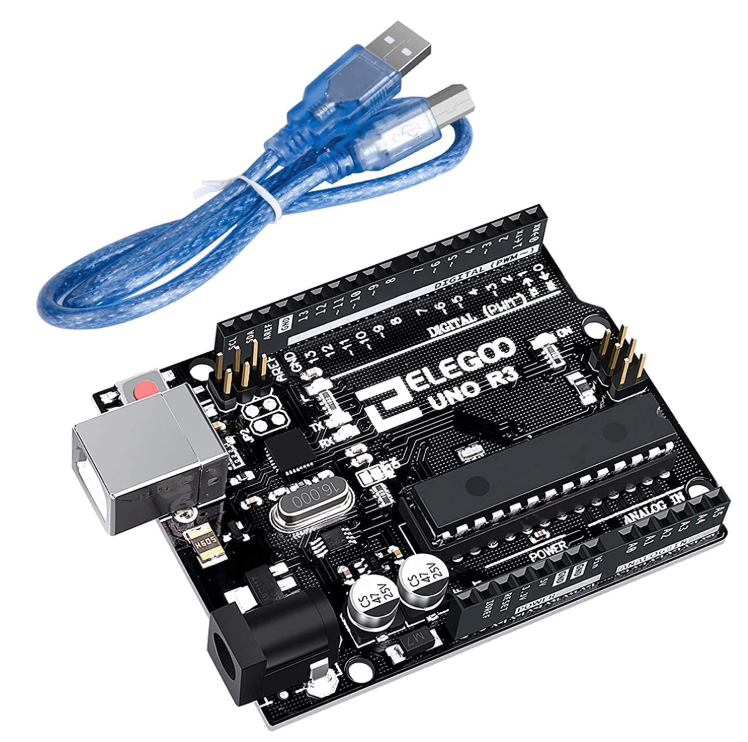
Here is an example of the LCD and GPS modules and a low cost Arduino Uno clone.

<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/>



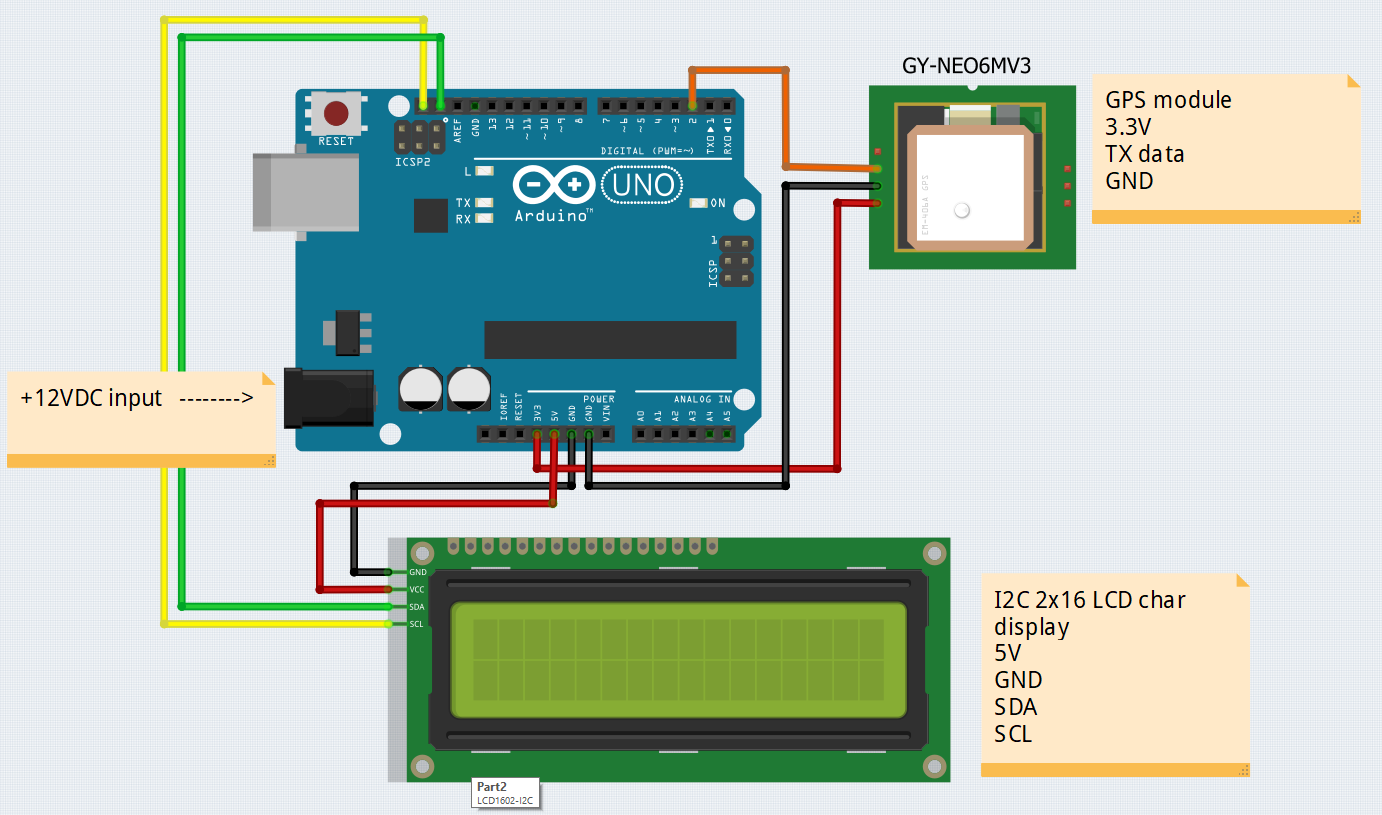
the github repository is located at <https://github.com/bpguy/gps_grid_sq_lcd> and contains the arduino sketch, this document, and the libraries needed (in case they are not easy to find.)

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.



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.

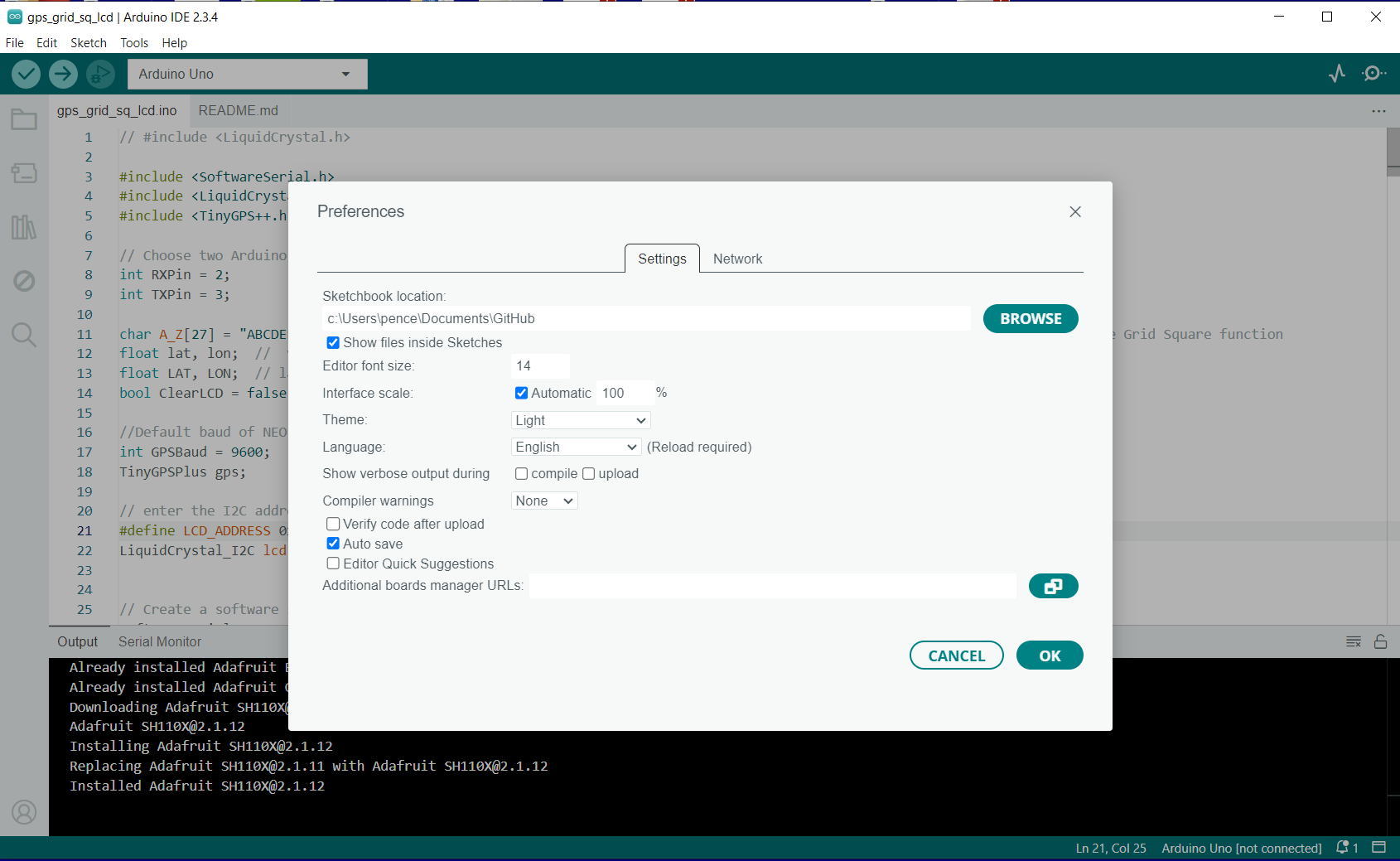
The arduino libraries needed are TinyGPSPlus and LiquidCrystal\_I2C (see notes below)

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

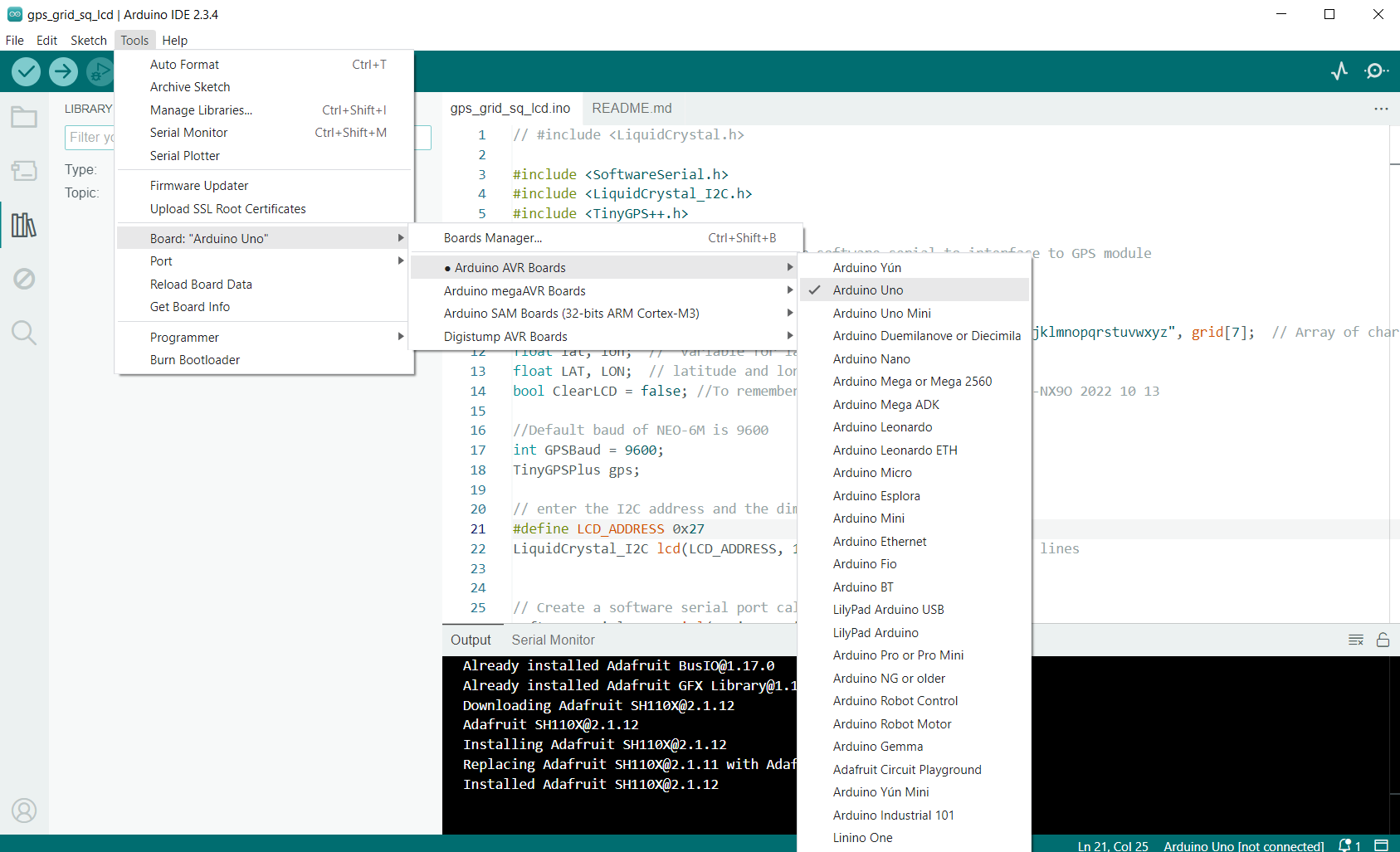
You can see the sketch folder location in the IDE menu

file -> preferences  settings.



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.

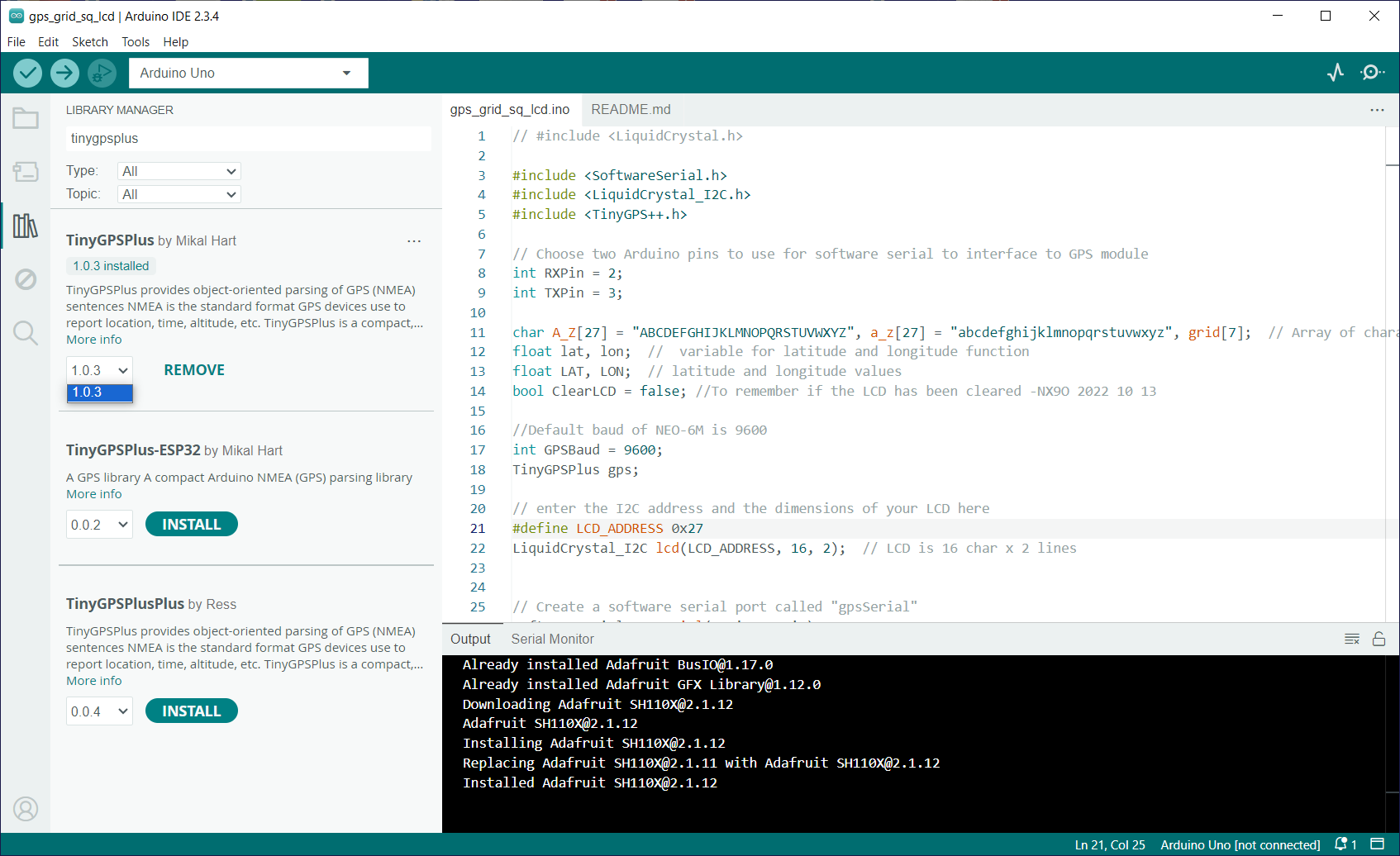
then open the sketch and select Arduino Uno (or the board you are using) from the board drop down.



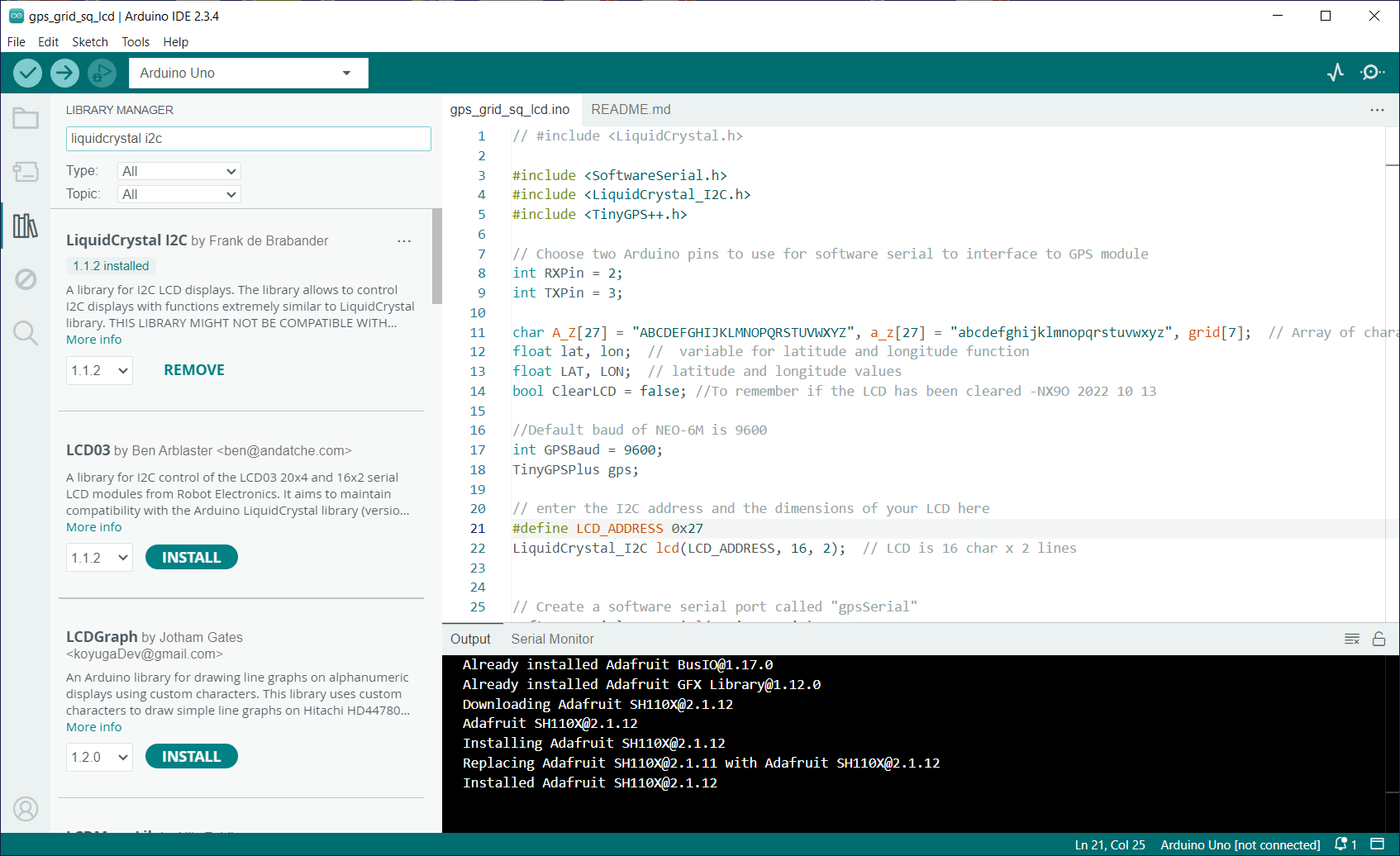
click the checkmark in the top left below “File” menu to try to compile. it should fail complaining about missing libraries.

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.



Then search LiquidCrystal I2C and install this one.



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;