```
/*
Talking UTC Clock/GPS/Grid Square Display by Glen Popiel - KW5GP

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 3 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

  Nokia 5110 LCD module is connected to the following pins.
      CLK  - Pin 12
      DIN  - Pin 11
      DC   - Pin 10
      CE   - Pin 9
      RST  - Pin 8
*/
#include <math.h>  // so we can get absolute value of floating point
number

#include <LCD5110_Basic.h>  // Include Nokia 5110 LCD Library

LCD5110 glcd(12,11,10,8,9);  // Define the LCD object

extern uint8_t SmallFont[];  // Define the LCD Font

#include <SoftwareSerial.h>  // include the SoftwareSerial library so we
can use it to talk to the Emic 2 module

#define emic_rxPin -1    // Serial input (connects to Emic 2 SOUT)
#define emic_txPin 3    // Serial output (connects to Emic 2 SIN)
#define audio_on 7    // Audio Enable on Pin 7

SoftwareSerial emicSerial =  SoftwareSerial(emic_rxPin, emic_txPin);  //
set up a new serial port

#define gps_rxPin 5    // Serial input (connects to GPS TxD)
#define gps_txPin -1    // Serial output (connects to GPS RxD) Set to -1
(disabled) because we don't need to send anything to GPS

#include <TinyGPS.h>  // Include the TinyGPS Library
#define gps_reset_pin  4  //GPS Reset control

SoftwareSerial GPS(gps_rxPin, gps_txPin);  // set up a new serial port

TinyGPS gps;  // set up a new GPS object
```

```
void gpsdump(TinyGPS &gps);  // Define the gpsdump function
bool feedgps();  // Define the feedgps function
void getGPS();  // define the getGPS function

long lat, lon;  // long integer for latitude and longitude function
float LAT, LON;  // floating integer for latitude and longitude values
int year;  // Variable to hold the year value
int gps_altitude;   // Variable to hold the altitude value
int gps_tick = 0, gps_timeout = 120;  // GPS startup timer variables
byte month, day, hour, minute, second;  // variables to hold date and
time
unsigned long fix_age, time, date, chars;
String s_date, s_time, s_month, s_year, s_day, s_hour, s_minute,
s_second, grid_text; // String variables for date, time and grid square
char A_Z[27] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ", a_z[27] =
"abcdefghijklmnopqrstuvwxyz", grid[7];  // Array of characters for the
Grid Square function
boolean gps_ok;  // gps data ok flag

void setup()
{
  // define the pin modes
  pinMode(emic_rxPin, INPUT);  // set up the serial I/O pins for the Text
to Speech module
  pinMode(emic_txPin, OUTPUT);
  pinMode(gps_rxPin, INPUT);// set up the serial I/O pins for the GPS
module
  pinMode(gps_txPin, OUTPUT);
  pinMode(audio_on, INPUT);  // Setup the Audio enable pin as an input
  digitalWrite(audio_on, HIGH);  // enable pullup resistor
Serial.begin(9600);
  emicSerial.begin(9600);  // set the data rate for the SoftwareSerial
port
  GPS.begin(9600);  // set the data rate for the SoftwareSerial port

  glcd.InitLCD();  // Initialize the Nokia 5110 Display
  glcd.setFont(SmallFont);  // Use the small font

  // Display the Startup screen
  glcd.clrScr();
  glcd.print("KW5GP", CENTER,0);
  glcd.print("UTC/GPS", CENTER,8);
  glcd.print("Grid Square", CENTER,16);
  glcd.print("Display", CENTER, 24);
  delay(3000);

  // Check for valid GPS data
  glcd.clrScr();
  glcd.print("GPS",CENTER,0);
  glcd.print("Acquiring Sats",CENTER,8);
  glcd.print("Please Wait",CENTER,32);

  // retrieves  lat/long in 100000ths of a degree
  gps.get_position(&lat, &lon, &fix_age);  // Read the GPS
```

```
  gps_ok = false;
  gps_tick = 0;
  while (fix_age == TinyGPS::GPS_INVALID_AGE & gps_tick < gps_timeout) //
Loop until we start getting valid GPS messages
  {
    gps.get_position(&lat, &lon, &fix_age);  // Read the GPS
    getGPS();
    glcd.print("No Sat. Fix", CENTER,16);
    glcd.print((" " String(gps_timeout - gps_tick)  " "),CENTER,40);  //
Display the timeout timer
    delay(1000);
    gps_tick;  // Wait a second and decrement the timeout timer
  }
  if (gps_tick < gps_timeout)  // Check to see if valid message
  {
    gps_ok = true;  // We got valid data before timeout, flag the GPS
data as valid
  }

  glcd.clrScr();

  // Set up the Text to Speech Module
  emicSerial.print('\n');             // Send a CR in case the system is
already up
  delay(500);                          // Short delay
  emicSerial.println("n6");   // Set voice to Voice 6
  delay(500);
  emicSerial.print('v');    // Set the volume to □
  emicSerial.println("18");
  delay(500);
  emicSerial.print("W");  // Set the speech speed to 200wpm
  emicSerial.println("200");
  emicSerial.flush();                 // Flush the receive buffer

}  // End Setup Loop

void loop()    // Start the Main Loop
{
 if (gps_ok)  // If we have valid GPS data
 {
   glcd.print("Lat:",0,0);  // Set up the Nokia Display with our data
template
   glcd.print("Lon:",0,8);
   glcd.print("Sats:",0,16);
   glcd.print("Date:",0,24);
   glcd.print("Time:",0,32);

   getGPS();  // Read the GPS
   gps.get_position(&lat, &lon, &fix_age);     // Read the GPS Latitude
and Longitude
   gps.get_datetime(&date, &time, &fix_age);  // Read the GPS Data and
Time

   s_date = String(date);  // convert the date and time to strings
```

```
   s_time = String(time);

   Serial.print(s_date); Serial.print("   ");
Serial.print(s_date.length()); Serial.print("  ");
   Serial.print(s_time); Serial.print("   ");
Serial.println(s_time.length());

   while (s_date.length() < 6)  // The conversion to string removes
leading zeroes, add them back
   {
      s_date = "0"  s_date;
   }

   while (s_time.length() <8)   // The conversion to string removes
leading zeroes, add them back
   {
     s_time = "0"  s_time;
   }

   s_year = s_date.substring(4,6);  // Break out the date string into
Day/Month/Year
   s_month = s_date.substring(2,4);
   s_day = s_date.substring(0,2);
   s_hour = s_time.substring(0,2);  // Break out the time string into
Hour/Minute/Second
   s_minute = s_time.substring(2,4);
   s_second = s_time.substring(4,6);

   glcd.printNumF(fabs(LAT/1000000),4,30,0,0x2e,7);  // Display the
Latitude and North/South
   if (LAT > 0)
   {
     glcd.print("N",78,0);
   } else if (LAT < 0)
   {
     glcd.print("S",78,0);
   }

   glcd.printNumF(fabs(LON/1000000),4,30,8,0x2e,7);  // Display the
Longitude and East/West
   if (LON > 0)
   {
     glcd.print("E",78,8);
   } else if (LON < 0)
   {
     glcd.print("W",78,8);
   }

   glcd.printNumI(int(gps.satellites()),36,16,2);  // Display the number
of Satellites we're receiving

   glcd.print(s_month  "/"  s_day  "/"  s_year,36,24);  // Display the
Date
```

```
   glcd.print(s_hour  ":"  s_minute  ":"  s_second,36,32);  // Display
the Time

   GridSquare(LAT,LON);  // Calulate the Grid Square

   glcd.print(grid_text,0,40);  // Display the Grid Square

   gps_altitude = int(gps.f_altitude());  // Read the GPS altitude
   glcd.print(String(gps_altitude)  "m  ",50,40);  // Display the
altitude

   if (digitalRead(audio_on) == LOW)  // Only speak if Audio is enabled
   {
     emicSerial.print("S");  // Say the Time
     emicSerial.print("U T C Time is:");
     emicSerial.print(s_hour  " "  s_minute);

     emicSerial.print(" Grid Square is:");  // Say the Grid Square one
letter at a time
     for (int x = 0; x <= 5; x)
     {
       emicSerial.print(grid_text[x]);
       emicSerial.print(" ");
     }

     emicSerial.print('\n');  // Send a carriage return to start speaking
   }
   } else
   {  // We did not get valid GPS before the startup timed out
     glcd.clrScr();
     glcd.print("NO GPS Data",CENTER,40);  // Display No GPS Data message
     if (digitalRead(audio_on) == LOW)  // Only speak if Audio is enabled
     {
       emicSerial.print("S");  // Say the Time
       emicSerial.print("U T C Time is Unknown");
       delay(1000);
       emicSerial.print("Grid Square is Unknown");
       emicSerial.print('\n');  // Send a carriage return to start
speaking
     }
   }




  delay(5000);

}  // End Main Loop

void getGPS()  // Function to get new GPS data every second
{
  bool newdata = false;
```

```
  unsigned long start = millis();
  // Every 1 seconds we print an update
  while (millis() - start < 1000)
  {
    if (feedgps ())
    {
      newdata = true;
    }
  }
  if (newdata)
  {
    gpsdump(gps);
  }
}

bool feedgps()  // Read the GPS data
{
  while (GPS.available())
  {
    if (gps.encode(GPS.read()))
    return true;
  }
  return 0;
}

void gpsdump(TinyGPS &gps)  //  Get the Latitude and Longitude
{
  gps.get_position(&lat, &lon);
  LAT = lat;
  LON = lon;
  {
    feedgps(); // If we don't feed the gps during this long routine, we
may drop characters and get checksum errors
  }
}

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;
  grid_text = " ";
  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□) - (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);
  grid[3] = (grid_lat_2  48);

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

  grid_text = grid;  // return the 6 character Gride Square

  return;

}
```