GPS Navigation für Geo-Cacher Teil 2

Hallo und Willkommen einer weiteren Folge unserer neuen GPS-Projektreihe. Im heutigen Teil machen wir aus unserem GPS Monitor ein waschechtes GPS-Navigationsgerät! Unser „Navi“ kennt zu jeder Zeit, wie auch alle modernen Navigationsgeräte, zwei unterschiedliche Geo-Koordinaten. Dies ist einmal die aktuelle Position [P], ermittelt aus den Moduldaten, und unsere Zielkoordinaten [T]. Aus diesen beiden Werten berechnet unser Navi die Entfernung in km und Meter, sowie die Himmelsrichtung zum Ziel. Diesen beiden Werten reichen aus um damit bis zum Zielpunkt zu navigieren!

Die Zielkoordinaten werden in den Zeilen:

```
float TargetGpsLAT = 48.85826;  // Coordinates EIFFEL_TOWER in France
float TargetGpsLON = 2.294516;
```

im Code deklariert. Im Code entsprechen sie den Koordinaten des Eifelturms in Frankreich. Diese können aber vor dem Hochlanden ganz einfach gegen eigene Koordinaten (zum Beispiel den eigenen Wohnort) ausgetauscht werden. Außerdem können die Ziel Koordinaten jederzeit durch Druck auf den Taster durch die aktuelle Position ersetzt werden. Nützlich ist dies Funktion z. B, wenn der Weg an einen bestimmten Ort zurückgefunden werden muss.

Wir brauchen für unser Navigationsgerät nur wenige Bauteile:

Anzahl	Bauteil	Anmerkungen
1	**HD44780 2004 LCD Display Bundle 4x20 Zeichen mit I2C Schnittstelle für Arduino**	
1	**NEO-6M GPS Modul**	
1	**Nano V3.0 CH340 Chip fertig verlötet**	
1	**Taster Modul**	

Alle notwendigen Informationen wie Orts und Geschwindigkeitsinformationen werden auf einem 4x20 Zeichen Display angezeigt.

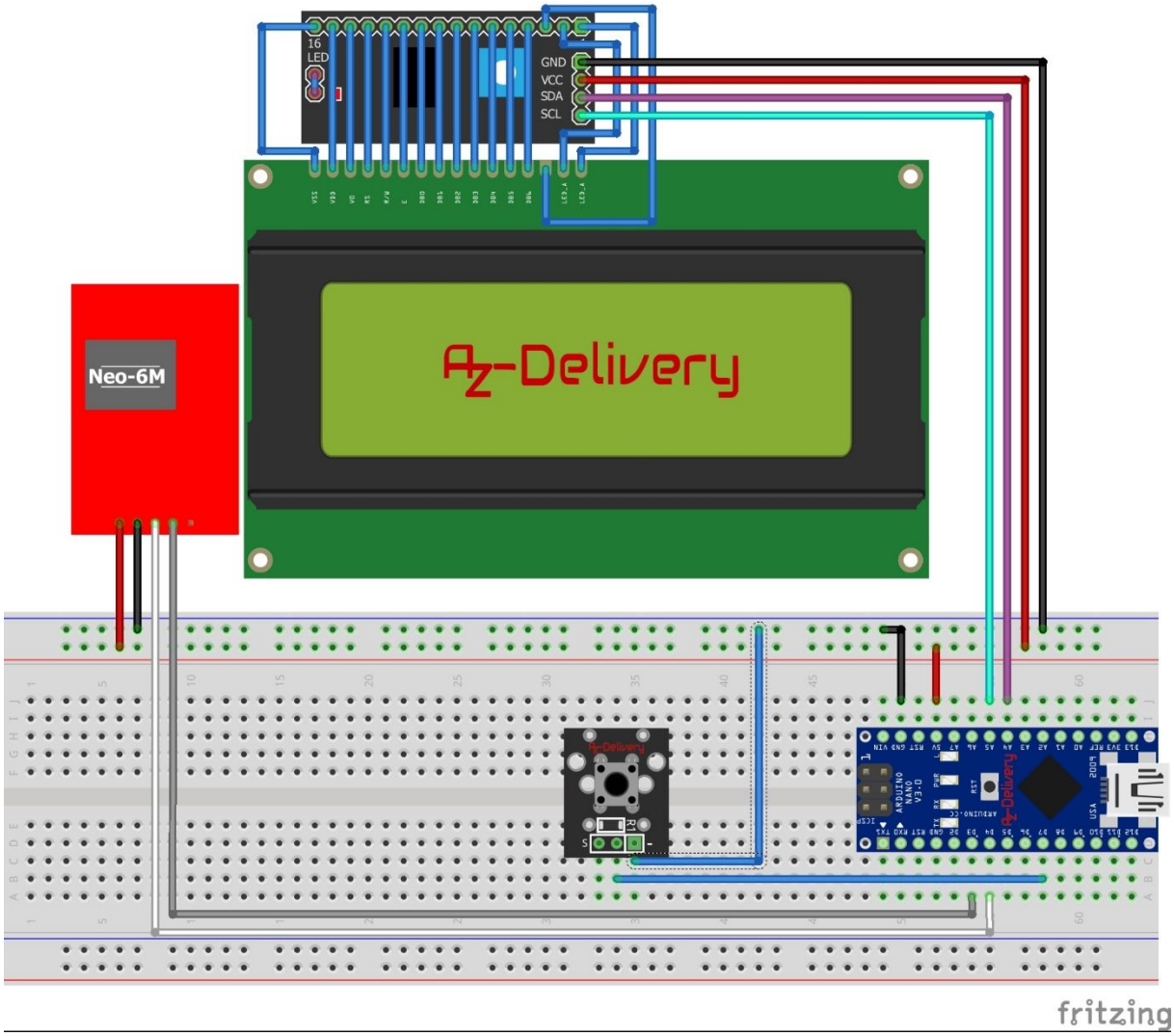
Kürzel	Bezeichnung	Ergänzende Hinweise
A	Altitude	Höhe über Wasser
Gs	Ground Speed	Akt. Geschwindigkeit
P	Position	Aktuelle Position
T	Target Position	Zielposition
W	Geographische Breite	Dezimale Schreibweise
L	Geographische Länge	Dezimale Schreibweise
Dst	Distance	Abstand zum Zielpunkt
Dir	Direction (Himmelsrichtung)	Richtung zum Zielpunkt

(Ansicht LCD-Display in Betrieb (fiktive Daten(Daten geändert))

Auch unser heutiges Projekt benötigt, wie das vorherige auch, zwei zusätzliche Bibliotheken. Die erste Bibliothek, TinyGPS ++ ist eine Arduino-Bibliothek zum Parsen von NMEA-Datenströmen, die von GPS-Modulen bereitgestellt werden. Diese Bibliothek bietet alle Methoden, die wir für unser Projekt benötigen. So nimmt Sie uns beispielsweise die Berechnungsarbeiten zur Entfernungsbestimmung zwischen aktuellem Ort und Zielort ab und stellt und diese in der Einheit Metern zur weiteren Bearbeitung zur Verfügung. Die Bibliothek TinyGPS++ kann auf GIT Hub heruntergeladen werden: https://github.com/mikalhart/TinyGPSPlus/releases

Die zweite, von unserem Projekt benötigte Bibliothek wird zur Ansteuerung des IC2 LCD-Displays benötigt. Die Bibliothek hört auf den Namen „LiquidCrystal I2C“ und ist von Frank de Brabander Sie kann direkt über den Bibliotheksverwalter unter dem gefunden und installiert werden:

Wie erweitern die Hardware um einen Taster, der die Zielkoordinaten aus den aktuellen Koordinaten übernimmt. Es kann somit jederzeit der aktuelle Standort aus Zielort übernommen werden.



Nach Umbau bzw. Ausbau der Hardware kann der aktualisierte bzw. erweiterte Code hochgeladen werden:

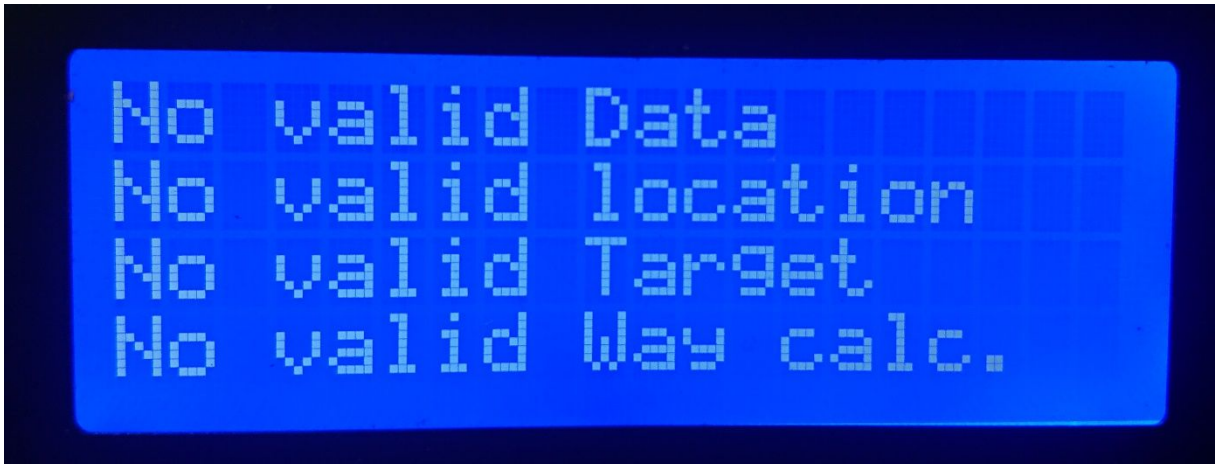
```
// GPS Empänger mit Navigation 2020 Tobias Kuch GPL 3.0
#define SwitchPIN 7
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 20 chars and 4 line display

TinyGPSPlus gps;
SoftwareSerial ss(4, 3);


float TargetGpsLAT = 48.85826;  // Coordinates EIFFEL_TOWER in France
float TargetGpsLON = 2.294516;
bool DistanceInKM = false;
bool Button1Pressed = false;
double Distance = 0;
```

```
double courseToT = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(SwitchPIN, INPUT_PULLUP); // Setzt den Digitalpin 13 als Outputpin
  lcd.init();                   // initialize theI2C lcd
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.clear();
  ss.begin(9600);
  smartdelay(1);
}

void loop()
{
if (gps.altitude.isValid())
    {
    if (gps.altitude.isUpdated()||gps.satellites.isUpdated()||gps.speed.isValid() )
      {
      lcd.setCursor(0,0);
      lcd.print("A:");
      lcd.print(gps.altitude.meters(),0); // Altitude in meters (double)
      lcd.print("m ");
      lcd.print("Gs:");
      lcd.print(gps.speed.kmph(),1);
      lcd.print("km/h");
      }
    } else
    {
    lcd.setCursor(0,0);
    lcd.print("No valid Data       ");
    }
if (gps.location.isValid())
    {
    if (gps.location.isUpdated())
      {
      lcd.setCursor(0,1);
      lcd.print("P W:");
      lcd.print(gps.location.lat(), 4); // Raw longitude in whole degrees
      lcd.print(" L:");
      lcd.print(gps.location.lng(), 4); // Raw longitude in whole degrees
      }
    } else
    {
    lcd.setCursor(0,1);
    lcd.print("No valid location   ");
    }

if (TargetGpsLAT < 90 & TargetGpsLON < 180  )
    {
    lcd.setCursor(0,2);
    lcd.print("T W:");
```

```
   lcd.print(TargetGpsLAT, 4); // Raw longitude in whole degrees
   lcd.print(" L:");
   lcd.print(TargetGpsLON, 4); // Raw longitude in whole degrees
   } else
   {
   lcd.setCursor(0,2);
   lcd.print("No valid Target     ");
   }

if (gps.location.isValid() & TargetGpsLAT < 90 & TargetGpsLON < 180 )
   {
    DistanceInKM = false;
    Distance =
gps.distanceBetween(gps.location.lat(),gps.location.lng(),TargetGpsLAT,TargetGpsLON);
    courseToT = gps.courseTo(gps.location.lat(),gps.location.lng(),TargetGpsLAT,TargetGpsLON);
    if (Distance > 1000)
     {
     Distance = Distance / 1000;
     DistanceInKM = true;
     }
    lcd.setCursor(0,3);
    lcd.print("Dst:");
    lcd.print(Distance);
    if(DistanceInKM)  {lcd.print("km ");} else { lcd.print("m "); }
    lcd.print("Dir:");
    lcd.print(gps.cardinal(courseToT));
   } else
   {
   lcd.setCursor(0,3);
   lcd.print("No valid Way calc.  ");
   }

smartdelay(5000);
}

static void smartdelay(unsigned long ms)
{
 unsigned long start = millis();
 do
 {
  if (!(digitalRead(SwitchPIN)))   // read the input pin
    {
    delay(100);   //debouncing
    if (!Button1Pressed)
     {
     Button1Pressed = true;
     TargetGpsLAT = gps.location.lat();
     TargetGpsLON = gps.location.lng();
     lcd.setCursor(0,2);
     lcd.print("T W:");
     lcd.print(TargetGpsLAT, 4); // Raw longitude in whole degrees
     lcd.print(" L:");
     lcd.print(TargetGpsLON, 4); // Raw longitude in whole degrees
```

```
      }
    }  else
    {
    Button1Pressed = false;
    }

   while (ss.available())
    gps.encode(ss.read());
  } while (millis() - start < ms);
}
```

Nach dem Einschalten wird ein angepasster Initialisierungsbildschirm angezeigt:



Nach Empfang eines validen GPS-Signals (blaue LED des GPS Moduls blinkt) werden Geographie und Navigationsinformationen angezeigt. Im nachfolgenden Beispiel (und auch im Code) sind als Zielkoordinaten (Target) die Geodaten des Eifelturms eingegeben worden (Zeile T). Aus der aktuellen Position (Zeile P) errechnet sich der Abstand (Distance) bis zum Ziel von 492,58 km in Richtung Süden (S)

(fiktive Daten / Daten geändert)

Ich wünsche viel Spaß beim Nachbau. Wie immer findet Ihr alle Projekte unter der GitHub Seite https://github.com/kuchto