Geofencing Alarmanlage (Teil 3 der GPS Modul Reihe)

Willkommen zu einem neuen, spannenden Teil unserer GPS-Monitor Reihe. Heute zeige ich eine weitere spannende Verwendungsmöglichkeit unseres GPS Moduls. Wir bauen mit Hilfe des GPS Modules eine "Alarmanlage" für Mobile Dinge, wie z.B. ein Motorrad, ein Auto ein Campingwagen oder das eigene Fahrrad. Das besondere an der Geofencing "Alarmanlage" ist, das Sie auf Grundlage des Geofencings arbeitet. Dazu schauen wir uns zunächst auf Wikipedia die Definition des Begriffs Geofencing an:

Aus Wikipedia:

Mit Geofencing (Kunstwort aus englisch geographic „geographisch“ und fence „Zaun“) wird das automatisierte Auslösen einer Aktion durch das Überschreiten einer gedachten Begrenzung auf der Erdoberfläche oder in der Luft[1] bezeichnet. In den meisten Fällen definiert die Begrenzung eine geschlossene Fläche, so dass zwischen innen und außen unterschieden werden kann. Beispielsweise kann beim Eintritt in die Fläche oder beim Verlassen der Fläche eine Benachrichtigung ausgelöst werden. Das beobachtete Objekt muss dazu in regelmäßigen Abständen seine Position senden oder die Abfrage seiner Position von außen ermöglichen. Diese Ortsbestimmung kann über das Mobilfunksystem auf Funkzellenebene oder koordinatenbezogen über ein Navigationssatellitensystem erfolgen.
Als Geofencing-Bereiche können Kreise oder Rechtecke definiert werden; durch Verwendung von Polygonzügen können aber auch komplexe Geometrien, beispielsweise administrative Grenzen, verarbeitet werden. Die Entscheidung, ob sich das beobachtete Objekt innerhalb oder außerhalb des vordefinierten Gebietes befindet, wird mit Hilfe eines Geoinformationssystems getroffen.
Die ausgelöste Aktion kann sich auf eine Benachrichtigung per E-Mail oder SMS beschränken, sie kann bei Fahrzeugen aber auch die Aktivierung einer Wegfahrsperre sein oder auch ein Warnsignal des Gerätes, das dem Anwender die Überschreitung des Bereichs signalisiert. Durch Verwendung von Daten weiterer Sensoren kann die Entscheidung über eine Alarmauslösung differenziert werden.
Je nach Komplexität der Aufgabe werden die Daten von Dritten erhoben und ausgewertet, was, insbesondere bei der Personenüberwachung, mit zusätzlichen datenschutzrechtlichen Fragestellungen verbunden ist.

Quelle: Wikipedia

Was bedeutet dies für die Arbeitsweise unserer „Alarmanlage“? Nun, dies bedeutet, dass unser System nicht, wie vielen anderen Systemen bei Erschütterung des zu sichernden Objektes oder bei Aktivierung eines elektrischen Verbrauchers auslöst, sondern erst! bei Hinausbewegen des gesicherten Objektes aus einem selbst definierten, runden "virtuellen Zaun" heraus. (Daher stammt auch der Begriff des Geofencings) Dieser virtuelle Zaun wird beim Aktivieren der Alarmanlage von dem Arduino um den aktuellen Standort des Objektes gelegt. D.h wird bei aktivierter „Alarmanlage“ erst beim Verlassen der AKTUELLEN Standortes aus einem selbst definierbaren Radius hinaus Alarm ausgelöst. Dieser Radius kann im Programm mit Hilfe der Variable

```
double GeofenceinMeters = 30;
```

definiert werden. Im folgend vorgestellten Quelltext ist diese Variable mit einem Radius von 30 Metern vordefiniert. Diese sollte aber nach den eigenen Bedürfnissen angepasst werden:

```
// GPS Alarmanlage 2020 GPL 3.0
#define SwitchPIN 9
#define RGBLED_Red 8
#define RGBLED_Green 7
#define RGBLED_Blue 6
#define Relais_Alarm 2
#include <SoftwareSerial.h>
#include <TinyGPS++.h>

TinyGPSPlus gps;
SoftwareSerial ss(4, 3);


float ActualGpsLAT = 0;
float ActualGpsLON = 0;
bool DistanceInKM = false;
bool Button1Pressed = false;
double Distance = 0;
bool systemarmed = false;
bool initsystemarmed = false;
double GeofenceinMeters = 30;

bool Engage_Alarm = false;
byte SmartdelayLoop = 0;

void setup()
{
  systemarmed = false;
  Serial.begin(9600);
  pinMode(SwitchPIN, INPUT_PULLUP); // Setzt den Digitalpin 13 als Outputpin
  pinMode(RGBLED_Red,OUTPUT);
  pinMode(RGBLED_Green,OUTPUT);
  pinMode(RGBLED_Blue,OUTPUT);
  pinMode(Relais_Alarm,OUTPUT);
  digitalWrite(Relais_Alarm, HIGH);
  digitalWrite(RGBLED_Red, LOW);
  digitalWrite(RGBLED_Green, LOW);
  digitalWrite(RGBLED_Blue, HIGH);
  delay(100);
  digitalWrite(RGBLED_Red, HIGH);
  digitalWrite(RGBLED_Green, LOW);
  digitalWrite(RGBLED_Blue, LOW);
  delay(100);
  digitalWrite(RGBLED_Red, LOW);
  digitalWrite(RGBLED_Green, HIGH);
  digitalWrite(RGBLED_Blue, LOW);
```

```
  ss.begin(9600);
  smartdelay(1000);
}

void loop()
{

if (gps.location.isValid()& ActualGpsLAT < 90 & ActualGpsLON < 180 )
    {
    if (gps.location.isUpdated())
      {
      if ((systemarmed) && (initsystemarmed) )
        {
        initsystemarmed = false;
        digitalWrite(RGBLED_Red, HIGH);
        digitalWrite(RGBLED_Green, LOW);
        digitalWrite(RGBLED_Blue, LOW);
        Serial.println (gps.location.lat());
        Serial.println (gps.location.lng());
        Serial.println("ValidGPS Coordinates transferred to Memory");
        ActualGpsLAT = gps.location.lat();  // Location is stored in Memory
        ActualGpsLON = gps.location.lng();  // Location is stored in Memory
        Serial.println (ActualGpsLAT);
        Serial.println (ActualGpsLON);
        }
      double Getlat = gps.location.lat();
      double Getlng = gps.location.lng();

      Distance = gps.distanceBetween(Getlat,Getlng,ActualGpsLAT,ActualGpsLON);
      Serial.print("Distance Calculated: ");
      Serial.print(Distance);
      Serial.println(" m");
      if ((systemarmed) && (!(initsystemarmed)) )
        {
        if ((Distance > GeofenceinMeters) && ((ActualGpsLAT != gps.location.lat()) ||
(ActualGpsLON != gps.location.lng()) ) )

          {
            // engage time Limited Alarm !
             Engage_Alarm = true;
             SmartdelayLoop = 0;
             Serial.println("Alarm Engaged, transferring new GPS Coordinates to
Memory.");
             ActualGpsLAT = gps.location.lat();  // Location is stored in Memory
             ActualGpsLON = gps.location.lng();  // Location is stored in Memory
             Serial.println (ActualGpsLAT);
             Serial.println (ActualGpsLON);
          }
        }
      }
```

```
    }

Alarm_Control();
smartdelay(5000);
}


void Alarm_Control()
{
if (Engage_Alarm)
  {
  if (SmartdelayLoop < 7)
    {
    digitalWrite(Relais_Alarm, LOW);

    } else
    {
    Engage_Alarm = false;
    digitalWrite(Relais_Alarm, HIGH);
    }
  } else
  {
  Engage_Alarm = false;
  digitalWrite(Relais_Alarm, HIGH);
  }
}
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;
        systemarmed = !systemarmed;
        if (systemarmed)
          {
          digitalWrite(RGBLED_Red, HIGH);
          digitalWrite(RGBLED_Green, LOW);
          digitalWrite(RGBLED_Blue, HIGH);
          initsystemarmed = true;
          Serial.println("Alarm System Activated with Init Coordinates:");
          Serial.println (gps.location.lat());
          Serial.println (gps.location.lng());

          } else
          {
          Serial.println("Alarm System DeActivated");
```

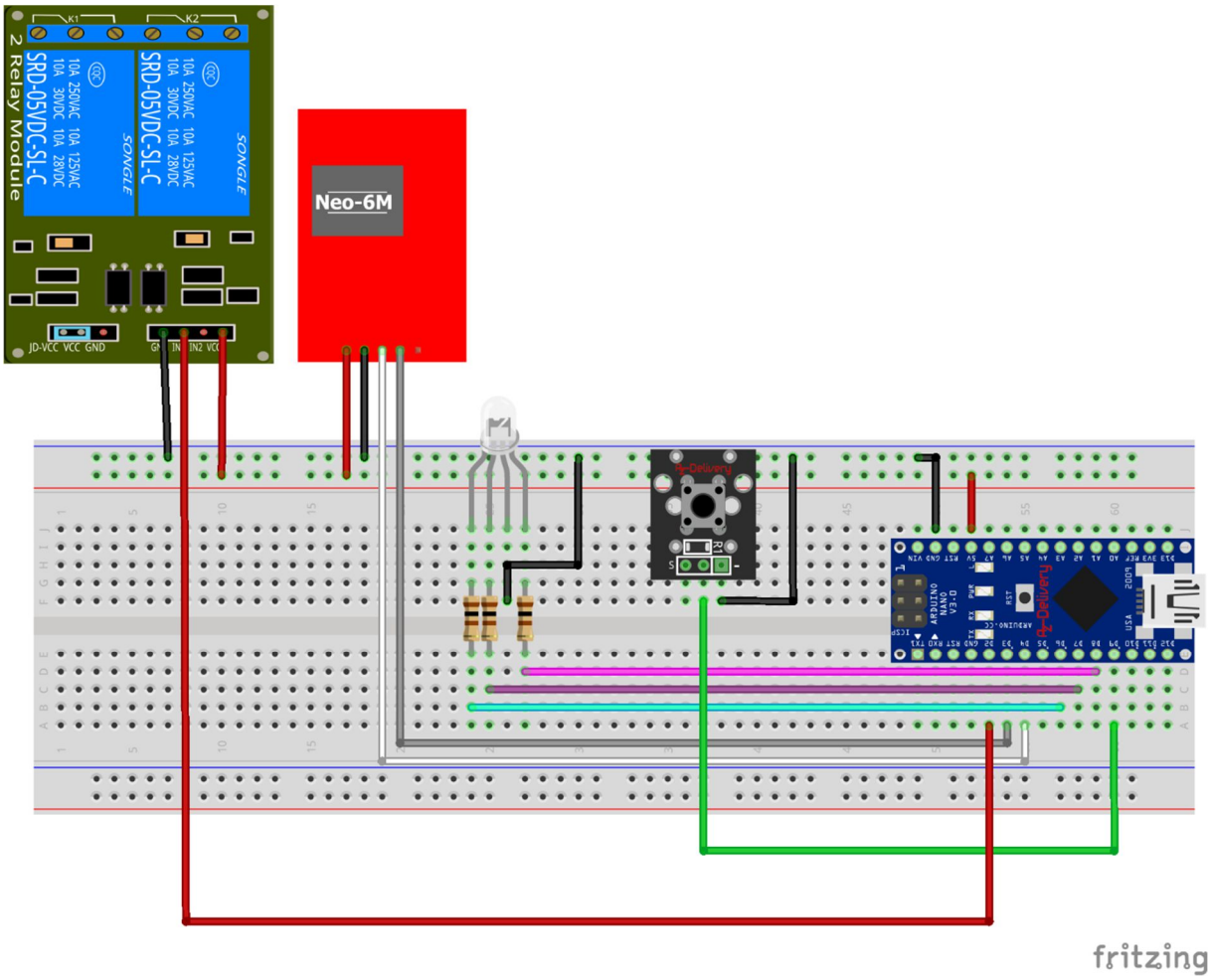
```
          digitalWrite(RGBLED_Red, LOW);
          digitalWrite(RGBLED_Green, HIGH);
          digitalWrite(RGBLED_Blue, LOW);
          initsystemarmed = false;
          Engage_Alarm = false;
          SmartdelayLoop = 0;
          digitalWrite(Relais_Alarm, HIGH);
          }
        }
      }  else
      {
      Button1Pressed = false;
      }

    while (ss.available())
      gps.encode(ss.read());
  } while (millis() - start < ms);
  SmartdelayLoop = SmartdelayLoop + 1;
}
```

Als Hardwareeinzelteile benötigen wir aus dem Shop:

Anzahl	Beschreibung	Zusatz
1	Ardunio Uno	
1	GPS Geo Modul	
1	Relais Modul	Low aktiv, 1 bis 2 Relais
1	Taster	
3	Wiederstände	120 Ohm
1	RGB LED	4 Pin

Wir verschalten die Hardware entsprechend dem Schemaplan:



Zur Bedienung und zur Kontrolle unserer „Geofence Alarmanlage“ stehen uns nachfolgende Bedienelemente zur Verfügung:

- **Status-Anzeige: RGB LED**
  Magenta: Die Alarmanlage ist scharf, Alarmanlagenfunktion ist jedoch noch nicht gegeben, da der GPS-Empfänger keine gültigen GPS-Informationen empfangen konnte. Eine Scharfschaltung erfolgt automatisch bei Empfang eines gültigen GPS-Signals.
  Grün leuchtend: Die Alarmanlage ist unscharf, die Alarmanlage kann bei Bedarf scharfgeschaltet werden.
  Rot leuchtend: Die Alarmanlage ist scharf.

- **Status-Änderung: Taster**
  Zur Scharfstellung bzw. Unscharf Stellung der Alarmanlage dienen die Port xx Eingänge des Prozessors. Durch kurzes Anlegen eines TTL Signals > 500 ms an einen der beiden Eingänge kann Über den einen kann unsere Alarmanlage aktiviert oder deaktiviert werden. Eingang Port : Taster Eingang: 5 Volt TTL Signal, LOW Aktiv: Alarmanlage scharf schalten (Zu beachten: Signal muss MINDESTENS 500 ms anliegen)Eingang Port : Taster Eingang: 5 Volt TTL Signal, LOW Aktiv: Alarmanlage unscharf schalten (Zu beachten: Signal muss MINDESTENS 500 ms anliegen)

Bei Auslösung der Alarmanlage wird der MOS-Transistor Ausgang (Port: 2 für 30 Sekunden auf low. Dieser Ausgang eignet sich für eine Sirene, die daran angeschlossen wurde, wird für diese für 30 Sekunden aktiviert.

Auf einen weiteren Schaltausgang, der bei Auslösung der Alarmanlage einen weiteren MOS-Transistor bis zur Deaktivierung der Alarmanlage aktiviert, wird aus guten Gründen! verzichtet.
Beispielsweise ist u.a ein zeitlich unbegrenzter und damit dauerhafter Alarm Ton in Deutschland nicht erlaubt!

Bitte beachtet, das das vorgestellte Projekt lediglich eine prinzipielle Durchführbarkeit belegt. Sollte diese Schaltung in der Praxis eingesetzt werden, schließe ich die Haftung für ewaitige daraus entstehende Schäden aus. Dies schließt auch genervte Nachbarn durch Falschalarme mit ein :)
Es soll hier lediglich das Prinzip des gefoncings anhand eines konkreten Anwendungsbeispiels verdeutlicht werden. Ich wünsche viel Spaß beim Nachbau und Experimentieren. Bis zum nächsten Mal.