Projekt modułu GPS zbudowanego na płytce Arduino Uno do śledzenia pozycji w aplikacji webowej opartej na PHP i MySql.

23.12.2019

Autor projektu: Kamil Piech 38586

w ramach kursu: Techniki mikroprocesorowe i systemy wbudowane
Prowadzący: mgr inż. Jan Duda

1. Wstęp

Założeniem projektu jest stworzenie modułu GPS, który ma wskazywać aktualną pozycję w aplikacji webowej. Większość nowoczesnych urządzeń takich jak smartfony czy tablety nam to umożliwiają poprzez wbudowany moduł GPS i takie aplikacje jak Google Maps do śledzenia aktualnej pozycji urządzenia. Poniżej wyjaśnienie pojęcia co to jest ten GPS.

Global Positioning System (**GPS**) – właściwie GPS–NAVSTAR (ang. Global Positioning System – Navigation Signal Timing and Ranging) – system nawigacji satelitarnej, stworzony przez Departament Obrony Stanów Zjednoczonych, obejmujący swoim zasięgiem całą kulę ziemską. System składa się z trzech segmentów: segmentu kosmicznego – 31 satelitów orbitujących wokół Ziemi na średniej orbicie okołoziemskiej; segmentu naziemnego – stacji kontrolnych i monitorujących na Ziemi oraz segmentu użytkownika – odbiorników sygnału. Zadaniem systemu jest dostarczenie użytkownikowi informacji o jego położeniu oraz ułatwienie nawigacji po terenie.

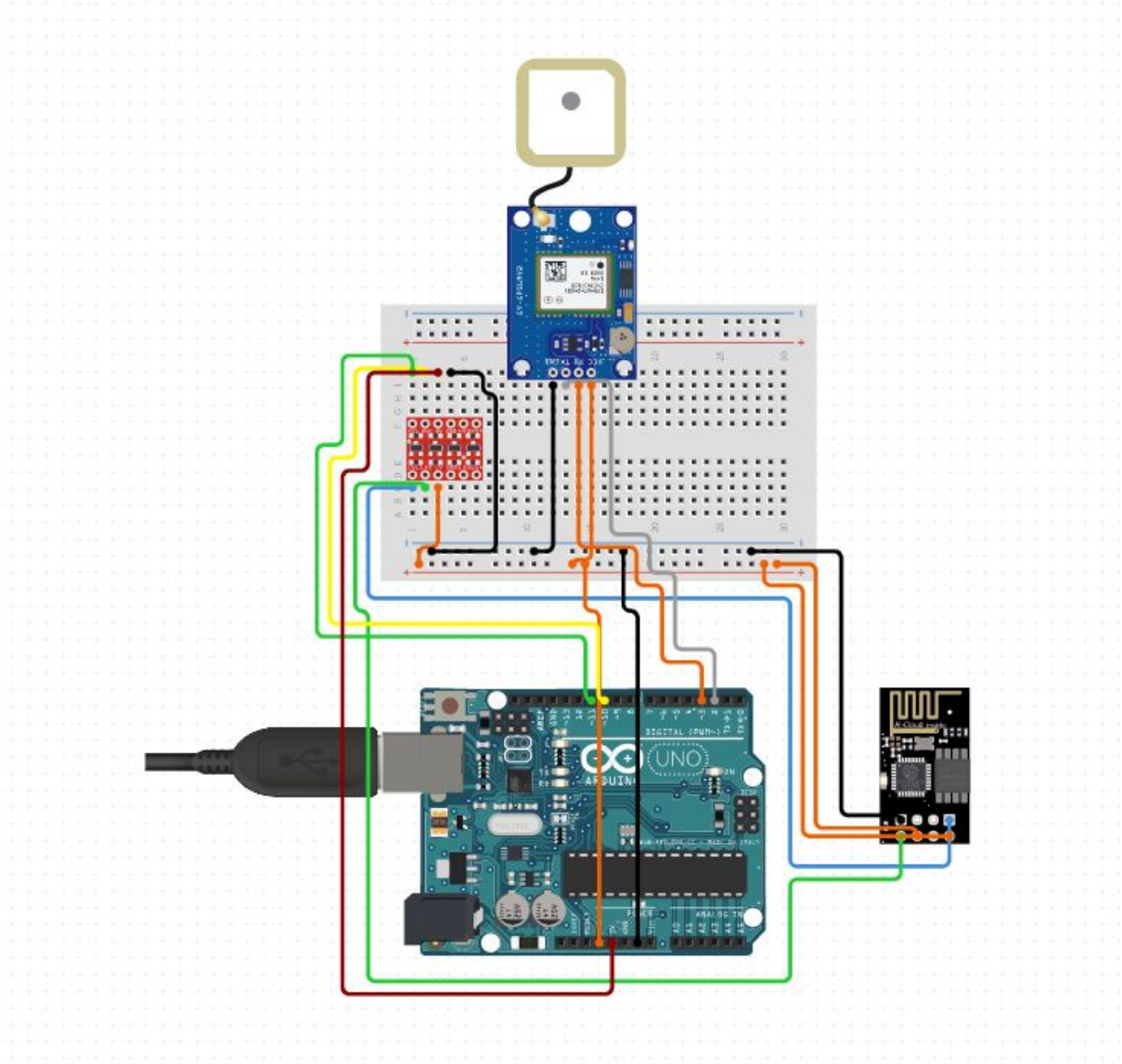
2. Cele projektowe

1. Implementacja podłączenia do sieci WIFI.
2. Implementacja modułu GPS.
3. Wysyłka zebranych danych z GPS na serwer.
4. Prezentacja danych w aplikacji webowej.

3. Podzespoły użyte do budowy projektu

1. Płytka Arduino - Arduino Uno R3.
2. Moduł GPS - GPS GY - NEO-6MV2.
3. Moduł WIFI - ESP8266 WiFi.

4. Schemat



5. Opis programu

Cały kod programu, lista zmiennych i wszystkie funkcje modułu dostępne są pod tym linkiem https://github.com/kamilpiech97/arduino-project.

1. Inicjowanie bibliotek oraz zmiennych

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>     // Include the Wi-Fi library

const char* ssid     = "Orange-7EFE";         // The SSID (name) of the Wi-Fi network you want to connect to
const char* password = "*********";     // The password of the Wi-Fi network

static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);
```

Bibliotek Tiny oraz SoftwareSerial to biblioteki potrzebne do inicjalizacji połączenia i zbierania danych z modułu GPS zaś ESP8266WiFi oraz ESP8266HTTPClient to biblioteki potrzebne do połączenia się z WiFi oraz wysyłki danych na serwer. Zmienne ssid oraz password są to dane autoryzacyjne służące do połączenia się z WiFi czyli interentem.

Reszta zmiennych to inicjalizacja portów modułu GPS.

2. Setup

```
void setup(){
  Serial.begin(9600);
  ss.begin(GPSBaud);

  WiFi.begin(ssid, password);             // Connect to the network
  Serial.print("Connecting to ");
  Serial.print(ssid);
  Serial.println("...");

  int i = 0;
  while (WiFi.status() != WL_CONNECTED) { // Wait for the Wi-Fi to connect
    delay(1000);
    Serial.print(++i); Serial.print(' ');
  }


  Serial.println('\n');
  Serial.println("Connection established!");
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());         // Send the IP address of the ESP8266 to the computer
}
```

W inicjalizacji naszej aplikacji po kolei idąc uruchamiamy czytanie danych z portu szeregowego następnie jest inicjowane połączenie z WiFi, następnie wykonuje się pętla while aż do uzyskania połączenia z WiFi i na samym końcu otrzymujemy informacje o adresie IP naszego WiFi z którym się połączyliśmy.

3. Loop

```
void loop(){
  // This sketch displays information every time a new sentence is correctly encoded.
  delay(2000);
  Serial.print(ss.available());
  while (ss.available() > 0){
    gps.encode(ss.read());
    if (gps.location.isUpdated()){
      // Latitude in degrees (double)
      Serial.print("Latitude= ");
      double lat = gps.location.lat();
      Serial.print(lat, 6);
      // Longitude in degrees (double)
      Serial.print(" Longitude= ");
      double lng = gps.location.lng();
      Serial.println(lng, 6);

         if(WiFi.status()== WL_CONNECTED){   //Check WiFi connection status

             String contentType = "application/x-www-form-urlencoded";

             HTTPClient http;    //Declare object of class HTTPClient
             http.begin("http://pur.przedprojekt.com/gps");      //Specify request destination
             http.addHeader("Content-Type", "application/x-www-form-urlencoded");  //Specify content-type header
             String DatatoSend ="";
             DatatoSend += "lng=";
             DatatoSend += (lng);
             DatatoSend += "&ltd=";
             DatatoSend += (lat);
             //char data = "lng=".lng."&ltd=".lat."";
             int httpCode = http.POST(DatatoSend);   //Send the request
             String payload = http.getString();                  //Get the response payload

             Serial.println(httpCode);   //Print HTTP return code
             Serial.println(payload);    //Print request response payload

             http.end();  //Close connection

           }else{
              Serial.println("Error in WiFi connection");
           }

            delay(15000);  //Send a request every 15 seconds

    }
  }
}
```

Pętla co dwie sekundy sprawdza dostępność połączenia naszego modułu GPS z pobliską stacją / anteną GPS, gdy takie połączenie nastąpi zostają pobrane dane szerokości i długości geograficznej. Następnie po pobraniu danych, zostaje sprawdzone połączenie z WiFi. Pozytywny wynik połączenia pozwala aplikacji na wysyłke danych na serwer, następnie zostaje wyświetlony response z serwera - pozwala to sprawdzić czy serwer przyjął dane czy nie. Po wysłaniu danych aplikacja ma piętnaście sekund przerwy między następną wysyłką porcji danych o naszej pozycji.

Serial.print(lat, 6) - użycie liczby sześć pozwala wyświetlenie liczby do sześciu miejsc po przecinku zamiast jej większej ilości.

http.begin("http://pur.przedprojekt.com/gps") - ta funkcja pozwala na okrśelenie serwera na jaki będziemy wysyłać nasze dane z modułu GPS.

http.addHeader("Content-Type", "application/x-www-form-urlencoded") - nagłówek, który określa typ przesyłanych danych.

String DatatoSend ="";
DatatoSend += "lng=";
DatatoSend += (lng);
DatatoSend += "&ltd=";
DatatoSend += (lat);
int httpCode = http.POST(DatatoSend) - zmienna DatatoSend przechowuje dane podczas wysyłki na serwer. Po stronie serwera są one przechwytywane jako dane wysłane metodą **POST**.

```php
public function index() {
    $insert['lng'] = $_POST['lng'];
    $insert['ltd'] = $_POST['ltd'];
    $this->back_m->insert('gps', $insert);

    return $this->output
    ->set_content_type('application/json')
    ->set_output(json_encode(array('message' => 'success')));
}
```

6. Podsumowanie

Stworzenie projektu było dobrą nauką obsługi wysyłki danych na serwer oraz przechwytywanie odpowiedzi od serwera (response) ale również obsługi modułu GPS, którego na laboratorium nie poznaliśmy. Było to również dobre wyzwanie pod względem instalacji odpowiednich sterowników dla płytki Arduino, ponieważ została zakupiona przeze mnie z innego źródła niż od producenta oraz instalacja odpowiednich bibliotek dla WiFi oraz GPS aby je uruchomić.

Chciałbym zauważyć że zakupiony moduł GPS posiada za słabą antenę oraz czas połączenia ze stacją zazwyczaj wynosi od 30 do 60 minut, więc przy większym projekcie należałoby zainwestować w mocniejszy moduł GPS oraz z pewnego źródła.

Reasumując, jestem zadowolony z wykonanego projektu pomimo kilku przeszkód udało mi się go pomyślnie zrealizować i stworzyć powyższą dokumentację.