

XXXII. Országos Tudományos Diákköri Konferencia (OTDK)
Fizika, Földtudományok és Matematika Szekció
Kolozsvár, 2015. április 15–18.

Elektromágneses jelforrások operatív bemérése (helymeghatározása) – könnyen kezelhető, hordozható berendezés

Szerző:

Lingvay Dániel

Babeş–Bolyai Tudományegyetem, Kolozsvár, Fizika Kar, fizika-informatika szak,
alapképzés, I. év

Témavezetők:

dr. Tunyagi Artúr egyetemi adjunktus,

Babeş–Bolyai Tudományegyetem, Kolozsvár, Fizika Kar, Magyar Fizika Intézet

Összefoglaló: A dolgozat célja egy elektromágneses jelforrások (rádióadók) operatív bemérésére alkalmas, intelligens, könnyen kezelhető, hordozható (kisméretű) berendezés kifejlesztése. A megépített berendezés tartalmaz: egy kisméretű, hangolható és direktív ($5\div 15^\circ$ nyílásszögű) mágneses antennát, egy szélessávú USB kimenetű és szoftvervezérlésű rádióvevőt (Funcube Dongle - 64MHz-től 1,7GHz-ig), egy digitális iránytűt, egy GPS-t (Arduino UNO R3 AVR típusú mikrovezérlő, mellyel a bemért irányok és koordináták a Google Maps-ra felvihetők és a bemért felszín megtekinthető) és egy SSD-s laptopot. A berendezés vezérlésére és a mérési adatok feldolgozására egy speciális szoftver lett kifejlesztve.

Kulcsszavak: elektromágneses jelforrás, helymeghatározás, antenna, szoftvervezérlésű rádióvevő, PC-laptop, digitális iránytű, GPS, mikrovezérlő

1. Bevezető

A XX. század egyik legjelentősebb műszaki megvalósítása az elektromágneses hullámok legkülönbözőbb alkalmazásainak a kifejlesztése és ezek alkalmazása (rádió és TV műsorszórás, adatátvitel / adatközlés – rádiókommunikáció, a szállítóeszközök hely és iránymeghatározása – rádiónavigáció stb. A mai, modern világ szinte elképzelhetetlen elektromágneses (rádió) jelgenerátorok (adók) és detektorok (vevők) nélkül.

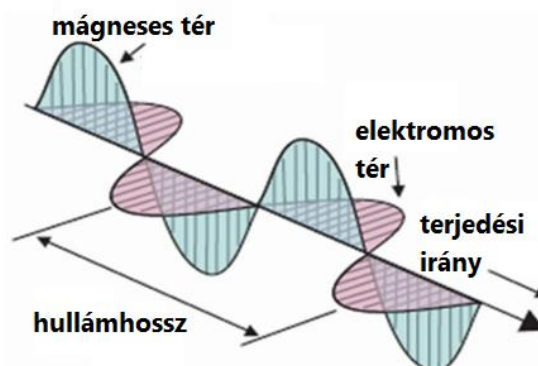
Az elektromágneses jelforrások (rádióadók) bemérése (helymeghatározása) egy nagy gyakorlati fontosságú témakör. A problematika először a hadászatban jelentkezett, amikor az ellenséges rádióadókat / adatközlőket kellett bemérni és semlegesíteni. Később a kalóz, hatósági engedély nélkül működő, műsorszóró rádióadók problémája is felmerült.

Napjainkban, az elektromágneses jelforrások helymeghatározásának egyik nagyfontosságú alkalmazása a bajbajutott szállítási eszközök (hajók, repülőgépek) valamint a GSM telefonkészülékek lokalizálása. Ezekre az alkalmazásokra komplex berendezések lettek kifejlesztve, amelyeknek kezelése megfelelő szaktudást igényel.

Ezekre való tekintettel, *a dolgozat célja*, egy az elektromágneses jelforrások (rádióadók, jeladók) operatív bemérésére alkalmas, intelligens, könnyen kezelhető (bárki által használható), hordozható (kisméretű), berendezés kifejlesztése.

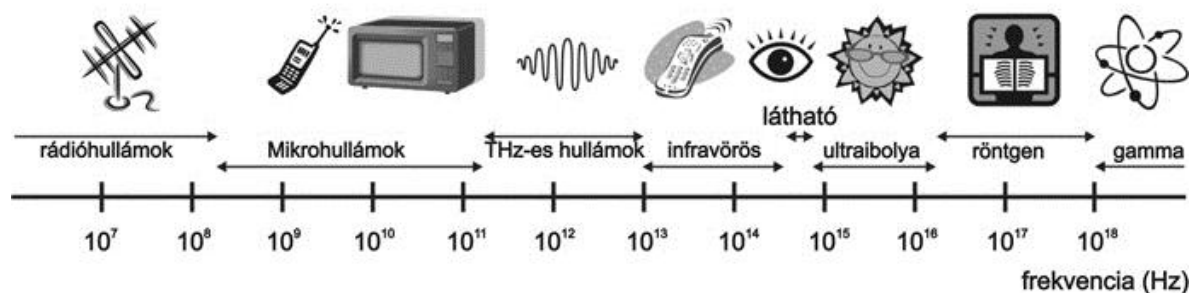
2. Elméleti alapok - az elektromágneses hullámok terjedése – rádiójelek vétele, antennák

Az elektromágneses sugárzás periodikusan változó elektromos és mágneses mezőkből áll – ezek merőlegesek a hullám haladási irányába (1. ábra).



1.ábra. Az elektromágneses hullámok

A hullámhossz, illetve a frekvencia ($\lambda = c/f$; $c = \text{fény sebessége vákuumban}$) függvényében az elektromágneses hullámok a következő csoportokra oszthatók (2. ábra):

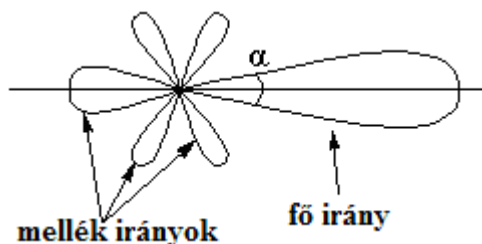


2. ábra. Elektromágneses hullámok csoportosítása

A hosszú, közép és ultrarövid rádióhullámokat főleg a hírközlésre (műsorszóró rádió és TV adók) használják. A mikrohullámokat főleg a speciális adatközlési rendszerekben alkalmazzák (GSM-mobiltelefon, digitális adatátvitel stb.). A rádióhullámok és a mikrohullámok frekvenciasávjainak használatát az ITU (International Telecommunication Union) által kidolgozott nemzetközi egyezmények szabályozzák.

Egy rádiókommunikációs rendszer főbb összetevői: a rádióadó és az adóantenna valamint a rádióvevő és a vevőantenna. A vevőantenna, tulajdonképpen egy elektromágneses hullámokat érzékelő szenzor – az elektromágneses hullámokat alakítja át elektromos jellé (feszültséggé) amelyet a vevőberendezés megfelelően feldolgoz.

A rádióhullámokat adó berendezések helymeghatározása elvileg egy kis nyílásszögű α Yagi típusú antennával és egy rádióvevővel lehetséges (3. ábra).



3. ábra. Az antenna iránykarakterisztikája

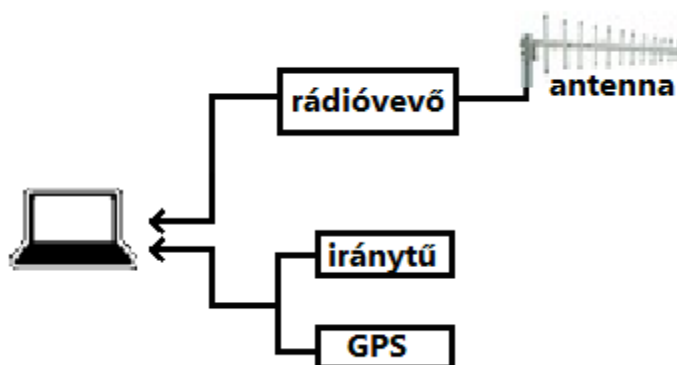
1.1. Az intelligens berendezés tervezése

A berendezés és a szoftver tervezésénél figyelembe vett főbb kritériumok:

- könnyen, szaktudást nem igénylő kezelhetőség – a berendezés kezeléséhez egy laptop általános használati ismerete elegendő legyen;
- kisméretű / hordozható – az egész berendezés tömege ne haladja meg a 2kg-ot, méretei (felülnézet) ne legyenek nagyobbak mint egy szabvány A4-es lap;

- működési autonómia (az akkumulátor újratöltése nélkül) – legalább 2 óra;
- a bemérhető jelek frekvenciasávja: - minim 64MHz – maxim 1,7GHz;
- a keresett jelforrás helymeghatározása – irány és az utolsó mérőponttól levő távolság – megfelelő (elfogadható) pontossággal;
- a meghatározott körzet megjelenítése (vizualizálása) Google Maps-on (ha van wireless internet jel);

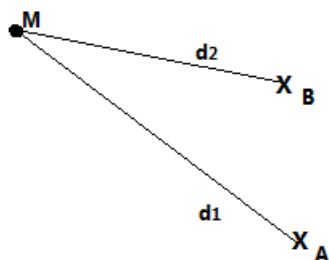
Ezekre való tekintettel, a megtervezett berendezés elvi vázlata a 4. ábrán van bemutatva.



4. ábra. Az intelligens berendezés elvi vázlata

A 3. ábrából megállapítható, hogy elvileg a berendezés tartalmaz egy direktív antennát, egy szélessávú rádióvevőt, egy hely és irány meghatározó egységet valamint egy adattároló és egy feldolgozó egységet (laptop). A meghatározás / rádióadó bemérés főbb lépései:

- egy kiinduló "A" mérőpont koordinátáinak meghatározása – a kapott adatok mentése;
- a keresett jelforrás (rádióadó) iránya az „A” mérőpontban – a kapott adat mentése;
- egy, az "A"-tól eltérő, "B" mérőpont koordinátáinak a meghatározása – a kapott adatok mentése;
- a keresett jelforrás (rádióadó) iránya a "B" mérőpontban – a kapott adat mentése;
- a mentett adatok feldolgozása – a d_1 és d_2 egyenesek "M" metszéspontjának a számítása/meghatározása – a koordináták kiírása;
- az utolsó mérőpont és az "M" pont közötti távolság számítása és kiírása.

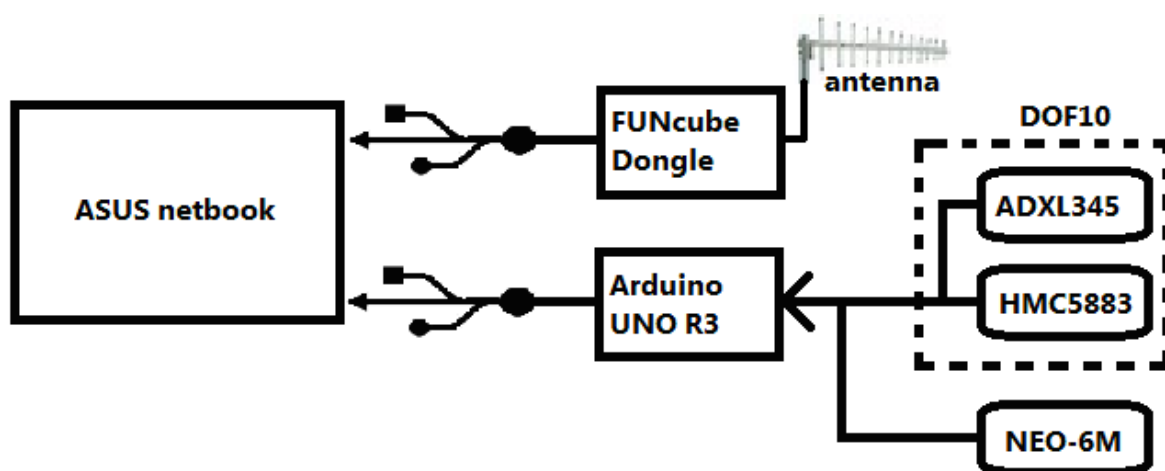


5. ábra. A helymeghatározás elvi vázlata

Az 5. ábrából megállapítható, hogy tekintettel az antenna irányjellemzőjére valamint a használt GPS pontosságára, a nagyobb pontosság érdekében, ajánlott a „B” mérőpontot az „A” –tól minél messzebbre választani (legalább a használt GPS pontossága +/- [m]-ben x 10) és lehetőleg a d_1 egyenesre merőleges irányban.

2. A berendezés kivitelezése

A 4. ábrán bemutatott vázlat alapján elkészített intelligens berendezés tömbvázlata a 6. ábrán van bemutatva.



6. ábra. A megépített berendezés tömbvázlata

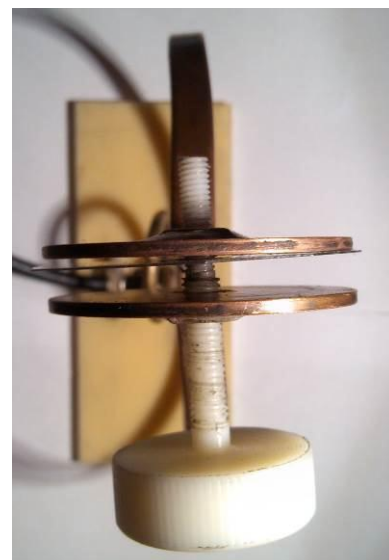
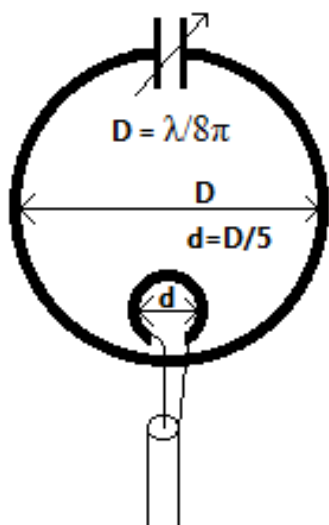
A 6. ábrából megállapítható, hogy a modulárisan kiképzett berendezés főbb elemei: egy kis nyílásszögű megfelelően megépített antenna; egy SDR (szélessávú) rádióvevő (Funcube Dongle), egy Arduino UNO „board” (AVR típusú fejlesztőplatform), GPS vevő (NEO-6M), egy mágneses tér szenzor / iránytű (HMC5883) és egy digitális gyorsulásmérő (ADXL345).

2.1. Az antenna kivitelezése

A helymeghatározáshoz használt antennának minél kisebb nyílásszöge (α) kell legyen. A gyakorlatban elterjedt YAGI típusú antennáknak habár kicsi az α -juk, viszont méretük aránylag nagy – a dipól hossza $\lambda/2$, ami például a 3m-es FM sávban ($88 \div 108$ MHz) kb. 1,5m-re adódik. Egy ilyen nagyméretű antenna nem felel meg a célnak (főleg erdős, bokros terepen). Erre való tekintettel kisméretű antennák lettek elkészítve.

Gyakorlatilag két antenna lett elkészítve. Egy egyszerűbb kivitelezésű mágneses, 90 és 130MHz között hangolható antenna (7 ábra), melynek iránykarakterisztikája a 8. ábrán van

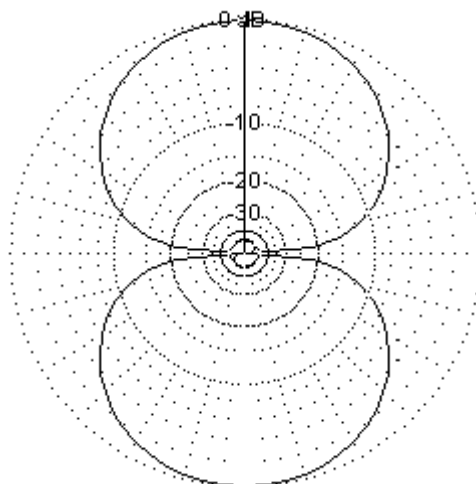
bemutatva, melyből megállapítható, hogy az irány karakterisztika hasonlít a szimpla dipóléhoz – csak az irány határozható meg vele, az irányítás nem.



7. ábra. A hangolható mágneses antenna elvi rajza és fényképe

A 7. ábrán bemutatott antenna működése azon alapszik, hogy a D átmérőjű tekercs a saját (szórt) valamint a C kapacitásokkal sorba kapcsolva egy adott frekvencián rezonál – amikor a nagy hurokban az áramerősség maximális, ami a C hangoló kapacitáson egy $U = L * \frac{dI}{dt}$ értékű feszültséget gerjeszt. A nagy hurok elektromágnesesen csatlakoztatva van egy kisebb hurokkal (tekercs) melynek végeire egy koaxiális kábel van kötve, ami a kisebb tekercsben gerjesztett feszültséget a rádióvevő bemenetére csatolja.

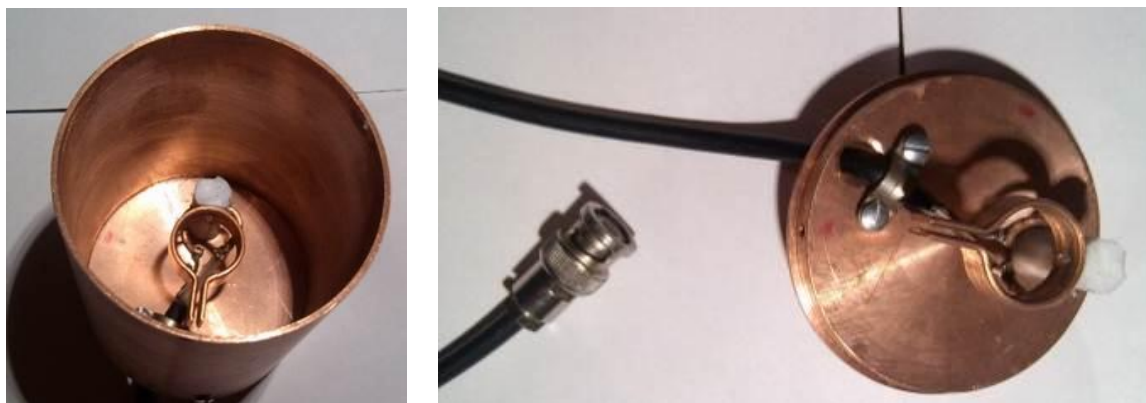
A hangoló C kondenzátor két párhuzamos lemezből lett kiképezve, az egyikben baloldali és a másikban jobboldali menetelésű lyukkal (részletek a 7. ábrán). A hangolás egy megfelelően menetelt szigetelő anyagból készült hangolótengely forgatásával történik – aminek hatására a két lemez közeledik, illetve távolodik egymástól.



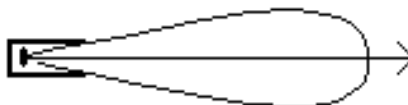
8. ábra. A 7. ábrán bemutatott antenna iránykarakteristikája.

A 8. ábrából megállapítható, hogy a mérések folyamán a keresett rádióadó iránya legpontosabban a vételi minimumon határozható meg (a nagy hurokkal egy síkban).

A második elkészített antenna a 800÷900MHz-es frekvenciasávban működő GSM telefonok bemérésére volt méretezve és a 9. ábrán van bemutatva. Elvileg, ugyancsak egy mágneses loop típusú antenna, viszont a kisebb α érdekében valamint az irányítás érdekében a két csatolt hurok egy rézcsőből készült árnyékoló serlegbe lett beépítve, ahova az elektromágneses hullámok csak az árnyékoló cső nyílásán juthatnak be. A 9. ábrán látható antenna iránykarakterisztikája a 10 ábrán van bemutatva.



9. ábra. A 800 ÷ 900MHz-es frekvenciasávra készült kis nyílásszögű antenna



10. ábra. A 9. ábrán bemutatott antenna iránykarakterisztikája

2.2. A rádióvevő egység

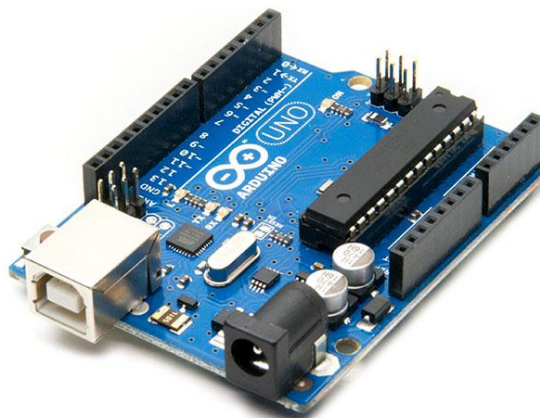
A rádióvevő egység egy FUNcube Dongle típusú SDR (software defined radio) amely USB porton keresztül van csatlakoztatva a számítógéphez. A vevő vezérlése / vételi frekvencia beállítása a HDSDR program segítségével történik 64 MHz és 1,7 GHz között 3,2 MHz-es sávszélességgel.



11. ábra. A használt rádióvevő és csatlakoztatása a laptop-hoz

2.3. A digitális adatfeldolgozó

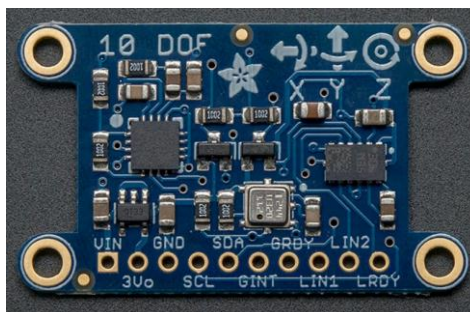
Adatfeldolgozóként egy Arduino UNO R3 board lett használva, mely az iránytűtől, gyorsulásmérőtől és a GPS-től kapott adatokat numerikusan feldolgozza, majd ezeket USB kábelén keresztül a laptopba táplálja. Erre külön program lett írva az Arduino programozási nyelvén, mely hasonló a C++-ra. A forráskód az 1. számú mellékletben van bemutatva.



12. ábra. A használt Arduino adatfeldolgozó fényképe

2.3.1. Iránytű

A berendezés egy DOF10 (mágneses tér szenzor, gyorsulásmérő, giroszkóp, hőmérő) lap (13. ábra), melyről két modul lett használva: egy HMC5883 típusú mágneses tér szenzor és egy ADXL345 típusú gyorsulásmérő. Mindkét modul a térbeli 3D vektor mindhárom komponenseit méri. A célnak, csak a mágneses tér szenzor nem elég, mivel az csak akkor ad jó eredményt, ha a lap (kb. 2cm x 1cm) pontosan vízszintes helyzetben áll. Ezt, a gyakorlatban, mérés közben elég nehéz megvalósítani, használni kellett egy gyorsulás mérőt is, mely megadja, hogy a **DOF10** lap mennyire van elfordulva a vízszintes helyzetéhez képest. A **DOF10** lap rá jön rögzítve az antennára, és az általa küldött adatok alapján a megírt program meghatározza a keresett jelforrás irányát. A modulok által adott adatokat egy egyszerű szűrőn keresztül vannak a programba táplálva, azért hogy a pillanatnyi mérési „zaj” ne befolyásolja a mérési eredményeket.



13. ábra. A DOF10 lap fényképe

2.3.2. A koordináták meghatározása - GPS

A GPS rendszer adás/vétel kapcsolatot egy U-BLOX NEO-6M modullal lett megoldva (14. ábra). Az U-BLOX NEO-6M modul által szolgáltatott adatbázisból az adatfeldolgozó kiszűri a földrajzi koordinátákat és ezeket betáplálja a számítógépbe. Az modul U-BLOX NEO-6M és az adatfeldolgozó 5 tizedes pontossággal határozzák meg a koordinátákat (ez a 45° északi szélességnél kb. 0,8m pontosságot biztosít).

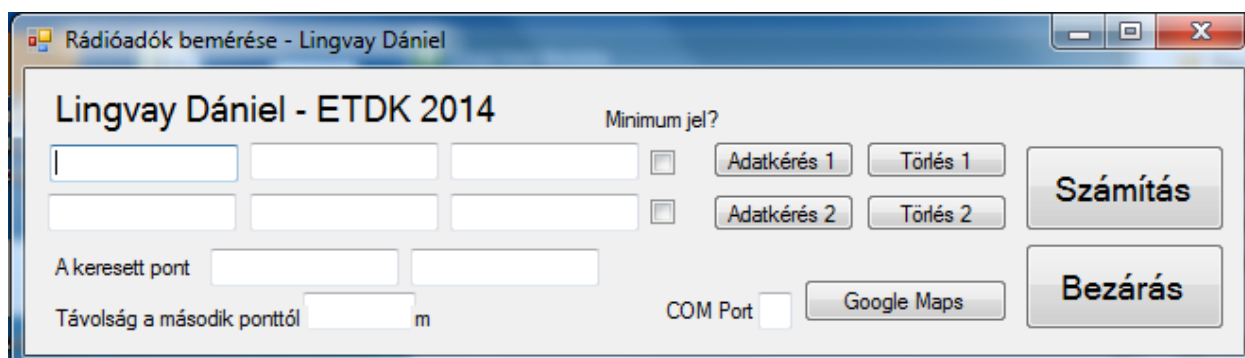


14. ábra. Az U-BLOX NEO-6M modul fényképe

2.4. A kifejlesztett szoftver és ennek futtatása

A kifejlesztett számítási szoftver C# programozási nyelvben lett megírva Windows operációs rendszerre. Ez azon alapszik, hogy adottak két mérőpont koordinátái valamint két irány a két mérőpontban meghatározott északhoz képest. Ezek alapján a program kiszámítja a két egyenes kereszteződési pontjának a koordinátáit valamint a távolságot a második mérőponttól a kereszteződési pontig (keresett jeladóig).

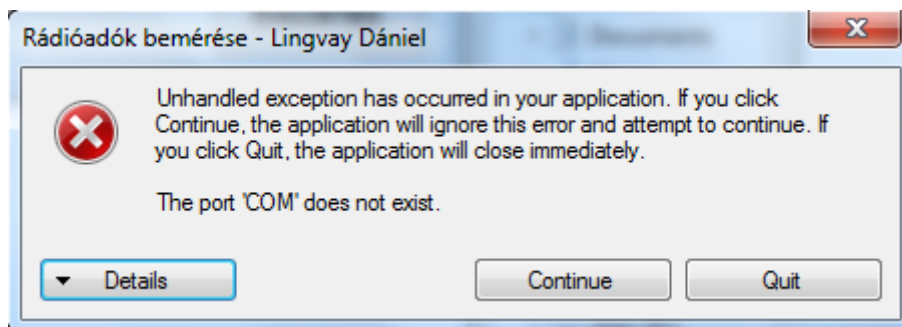
A program futtatásához NET Framework 2.0 szükséges, mely Windows XP, Windows 7 és Windows 8 telepítésénél automatikusa telepítődik. A program alapértelmezett kinézete a 15. ábrán van bemutatva.



15. ábra. A program alapértelmezett kinézete

A program használata:

A szoftver indítása után, az ablak alsó részében található *COM Port* szövegdobozba be kell írni a Windows által adott COM portot (Device Manager – Ports(COM & LPT) – Arduino Uno. Ha ez nincs helyesen megadva, a szoftver hibát ad (16. ábra.).

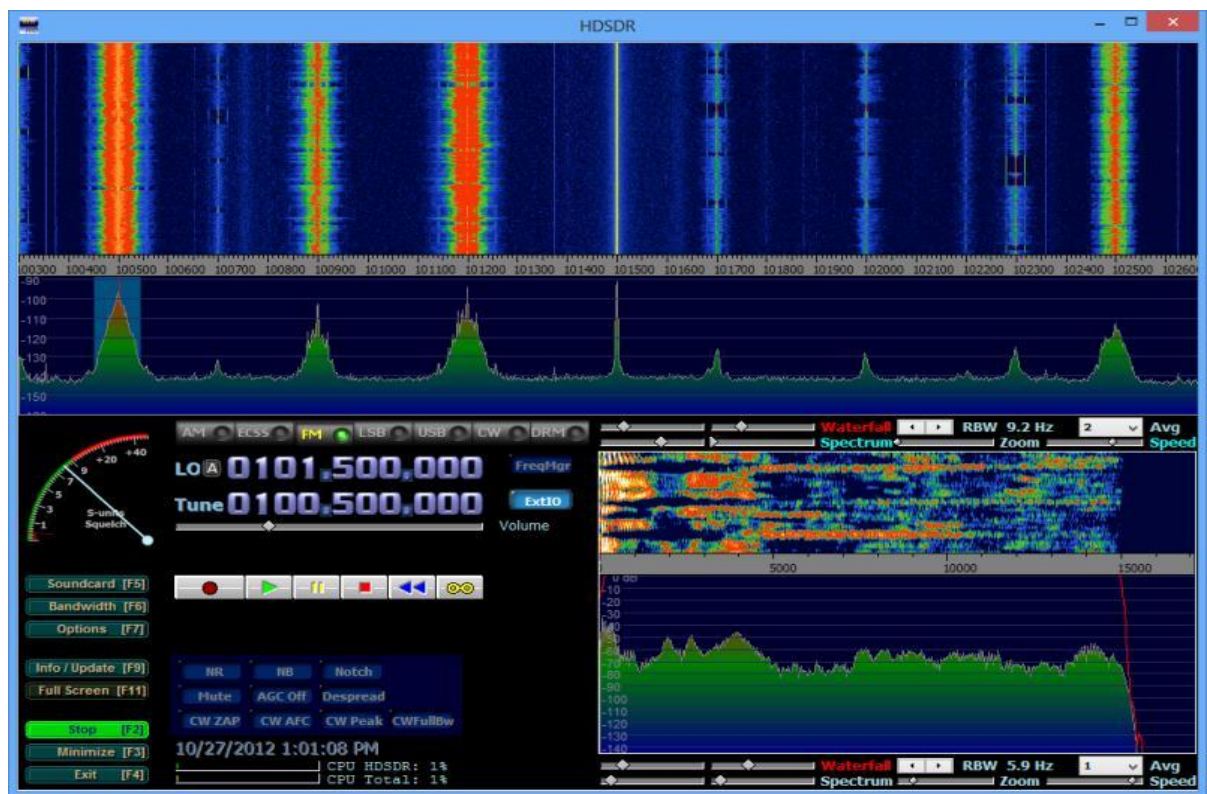


16. ábra. Futtatási hiba helytelen COM Port megadása esetén

A szoftver adatot kér az Arduino adatfeldolgozótól, ha az *Adatkérés 1* illetve *Adatkérés 2* ablak van aktiválva. Ha a leírásban bemutatott antennák közül a 7. ábrán bemutatottat használjuk (az irány jelminimumra jön beállítva), ki kell pipálni az adott sorban levő *jelölőnégyzetet* („COM Pot”). Ez hozzáad a kapott irányhoz 90⁰-ot, hogy a szoftver helyesen számoljon (jelmaximumra). Ha mindkét adat megvan, a *Számítás* gombbal indítjuk a számítást. A kapott adatok a 3 alsó szövegdobozba íródik ki (bal felső - szélességnél, jobb felső – hosszúság, alsó – távolság a második ponttól).

A *Google Maps* gombbal kinyílik egy új ablak, mely internet kapcsolat esetén betölti a Google Maps-et és egy *marker*-t tesz a keresett adó földrajzi helyzetében – a „Műhold” opcióval a körzetet vizualizálja, vagy a „Forgalom” opcióval a legkönnyebb megközelítési lehetőség választható ki.

A HDSDR program telepítése után és a Funcube Dongle driver-einek telepítése után be kell állítani a hangkártyát (F5 -> RX Input (from Radio)).



17. ábra. A HDSDR program fő ablaka – a felső képen: 91,625733 MHz-en – ahol nincs vehető adó, az alsó képen 100,5 és 101,5MHz között több adó vehető

A beállítás után az F2 gomb nyomásával indíthatjuk a vételt. Be kell állítani a frekvenciát (LO). Ez után a jel látható lesz az ablakban (17. ábra). Ez után változtatni kell a demodulátor modulációját a keresett jel függvényében.

3. Következtetések

Az elektromágneses jelforrások (rádióadók) operatív bemérésére alkalmas, intelligens, könnyen kezelhető, hordozható (kisméretű) berendezés lett kifejlesztve, megépítve és kipróbálva. A megépített berendezés tartalmaz: egy kisméretű, hangolható és direktív ($5\div 15^\circ$ nyílásszögű) mágneses antennát, egy szélessávú USB kimenetű és szoftvervezérlésű rádióvevőt (Funcube Dongle - 64MHz-től 1,7GHz-ig), egy digitális iránytűt, egy GPS-t (Arduino UNO R3 AVR típusú mikrovezérlő, mellyel a bemért irányok és koordináták a Google Maps-ra felvihetők és a bemért felszín megtekinthető) és egy SSD-s laptopot. Az adatfeldolgozó egység vezérlésére Arduino programozási nyelvben lett megírva a szoftver. A mérési adatok feldolgozására a szoftver C# programozási nyelvben lett megírva Windows operációs rendszerre.

A modulárisan kiépített ki lett próbálva és megfelelően működik. Ha a mérési helyen internet elérhetőség van, a GOOGLE Maps-on vizualizálja a keresett rádióadó (jeladó) körzetét valamint a megközelítési lehetőségeket.

A rendszer egy ASUS Netbook-ra (Windows operációs rendszerre) lett kifejlesztve, de a leírtak alapján megépíthető egy aránylag könnyebben kezelhető, Android operációs rendszerrel működő táblagépre is.

4. Irodalomjegyzék

- [1] ***** NEO-6M U-blox 6 GPS Modules - Data Sheet,
[https://www.u-blox.com/images/downloads/Product_Docs/NEO-6_DataSheet_\(GPS.G6-HW-09005\).pdf](https://www.u-blox.com/images/downloads/Product_Docs/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf)
- [2] ***** 10-DOF angle and altitude visualisation,
http://www.youtube.com/watch?v=DtUdlfKq6_U
- [3] Cristian Colonati, Alimentarea antenelor magnetice,
<http://www.radioamator.ro/articole/view.php?id=952>
- [4] Cristian Colonati, Dimensionarea antenelor magnetice folosite multiband,
<http://www.radioamator.ro/articole/view.php?id=941>

5. Mellékletek.

5.1. Az adatfeldolgozóra írt program forráskódja

```
#include "Wire.h"
#include "ADXL345.h"
#include "HMC5883L.h"
#include <SoftwareSerial.h>
#include <TinyGPS.h>

SoftwareSerial mySerial(10, 11); // TX, RX
TinyGPS gps;
char message;

const float alpha = 0.9;

double fXg = 0;
double fYg = 0;
double fZg = 0;

double pitch, roll, Xg, Yg, Zg, fi, teta;
double xdir, ydir, zdir;

int error = 0;
float heading;
double xh, yh, headingcompensated;

ADXL345 acc;
HMC5883L compass;

void setup()
{
  Serial.begin(57600);
  mySerial.begin(9600);
  acc.begin();
  Wire.begin();
  compass = HMC5883L();
```

```
error = compass.SetScale(1.3);  
error = compass.SetMeasurementMode(Measurement_Continuous);  
delay(100);  
}
```

```
void loop()  
{  
  if (Serial.available() > 0)  
  {  
  }  
  GPS();  
  ReadADXL();  
  ReadHMC();  
  tiltcompensate();  
  PrintCCompass();  
  delay(100);  
}
```

```
void ReadHMC()  
{  
  MagnetometerScaled scaled = compass.ReadScaledAxis();  
  xdir = scaled.XAxis;  
  ydir = scaled.YAxis;  
  zdir = scaled.ZAxis;  
  heading = atan2(ydir, xdir);  
  if(heading < 0)  
    heading += 2*PI;  
  if(heading > 2*PI)  
    heading -= 2*PI;  
  heading = heading * 180/M_PI;  
}
```

```
void ReadADXL()  
{  
  acc.read(&Xg, &Yg, &Zg);
```

```

fXg = Xg * alpha + (fXg * (1.0 - alpha));
fYg = Yg * alpha + (fYg * (1.0 - alpha));
fZg = Zg * alpha + (fZg * (1.0 - alpha));
pitch = asin(-fXg);
roll = asin(fYg/cos(pitch));
}

```

```

void PrintCCompass()
{
    int headingcompensatedi = (int)headingcompensated;
    if (headingcompensatedi < 100)
    {
        if (headingcompensatedi < 10)
        {
            Serial.print("00");
            Serial.println(headingcompensatedi);
        }
        else
        {
            Serial.print("0");
            Serial.println(headingcompensatedi);
        }
    }
    else Serial.println(headingcompensatedi);
}

```

```

void tiltcompensate()
{

    xh = xdir* cos(pitch) + zdir*sin(pitch);
    yh = xdir*sin(roll)*sin(pitch) + ydir * cos(roll) - zdir * sin(roll) * cos(pitch);
    headingcompensated = atan2(yh, xh);
    //if (yh < 0) headingcompensated += 360;
    if(headingcompensated < 0)
        headingcompensated += 2*M_PI;
}

```

```
if(headingcompensated > 2*M_PI)
    headingcompensated -= 2*M_PI;
headingcompensated = headingcompensated * 180 / PI;
}
```

```
void GPS()
{
    bool newdata = false;
    if (mySerial.available())
    {
        char c = mySerial.read();
        if (gps.encode(c))
        {
            newdata = true;
        }
    }
}
```

```
if (newdata)
{
    gpsdump(gps);
}
}
```

```
void gpsdump(TinyGPS &gps)
{
    long lat, lon;
    gps.get_position(&lat, &lon);
    delay(100);
    Serial.print(lat);
    Serial.print(lon);
}
```

A helymeghatározásra kifejlesztett szoftver forráskódja

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Diagnostics;
using System.IO;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        SerialPort serialPort1;
        char[] result = new char[19];
        string lat, lon, bearing;
        double lat1, lat2, lon1, lon2;
        double brng1, brng2;
        double lat3, lon3;

        public Form1()
        {
            InitializeComponent();
            serialPort1 = new SerialPort();
            serialPort1.BaudRate = 57600;
        }

        public char[] readserial()
        {
            for (int len = 0; len < result.Length; )
```

```

    {
        len += serialPort1.Read(result, len, result.Length - len);
    }
    return result;
}

```

```

private void fillbox1(string s)
{

    lat = s.Substring(0, 7);
    lon = s.Substring(7, 7);
    bearing = s.Substring(13);
    lat1 = Convert.ToDouble(lat) / 100000;
    lon1 = Convert.ToDouble(lon) / 100000;
    brng1 = Convert.ToDouble(bearing);
    TBlat1.Text = lat1.ToString();
    TBlon1.Text = lon1.ToString();
    TBdir1.Text = brng1.ToString();
}

```

```

private void fillbox2(string s)
{
    lat = s.Substring(0, 7);
    lon = s.Substring(7, 7);
    bearing = s.Substring(13);
    lat2 = Convert.ToDouble(lat) / 100000;
    lon2 = Convert.ToDouble(lon) / 100000;
    brng2 = Convert.ToDouble(bearing);
    TBlat2.Text = lat2.ToString();
    TBlon2.Text = lon2.ToString();
    TBdir2.Text = brng2.ToString();
}

```

```

private void button1_Click(object sender, EventArgs e)
{

```

```

serialPort1.PortName = "COM" + textBox2.Text;
serialPort1.Open();
if (serialPort1.IsOpen)
{
    serialPort1.WriteLine("A");
}
string s = new string(readserial());
fillbox1(s);
serialPort1.Close();
}

private void button2_Click(object sender, EventArgs e)
{
    serialPort1.Open();
    if (serialPort1.IsOpen)
    {
        serialPort1.WriteLine("A");
    }
    string s = new string(readserial());
    fillbox2(s);
    serialPort1.Close();
}

private void button3_Click(object sender, EventArgs e)
{
    this.Close();
}

public double ToRadian(double orig)
{
    orig = orig * Math.PI / 180;
    return orig;
}

public double ToDegree(double orig)

```

```

{
    orig = orig * 180 / Math.PI;
    return orig;
}

```

```

public void CalculateIntersection()
{
    double brng12, brng21;
    double lat1r = ToRadian(lat1);
    double lat2r = ToRadian(lat2);
    double lon1r = ToRadian(lon1);
    double lon2r = ToRadian(lon2);
    double brng13 = ToRadian(brng1);
    double brng23 = ToRadian(brng2);
    double dLat = lat2r - lat1r;
    double dLon = lon2r - lon1r;
    double dist12 = 2 * Math.Asin(Math.Sqrt(Math.Sin(dLat / 2) * Math.Sin(dLat /
2) + Math.Cos(lat1r) * Math.Cos(lat2r) * Math.Sin(dLon / 2) * Math.Sin(dLon / 2)));
    if (dist12 == 0) label3.Text = "Ugyan azok a pontok";
    // initial/final bearings between points
    double brngA = Math.Acos((Math.Sin(lat2r) - Math.Sin(lat1r) *
Math.Cos(dist12)) / (Math.Sin(dist12) * Math.Cos(lat1r)));
    if (double.IsNaN(brngA)) brngA = 0; // protect against rounding
    double brngB = Math.Acos((Math.Sin(lat1r) - Math.Sin(lat2r) *
Math.Cos(dist12)) / (Math.Sin(dist12) * Math.Cos(lat2r)));
    if (Math.Sin(lon2 - lon1) > 0)
    {
        brng12 = brngA;
        brng21 = 2 * Math.PI - brngB;
    }
    else
    {
        brng12 = 2 * Math.PI - brngA;
        brng21 = brngB;
    }
}

```

```
}
```

```
double alpha1 = (brng13 - brng12 + Math.PI) % (2 * Math.PI) - Math.PI; //  
angle 2-1-3
```

```
double alpha2 = (brng21 - brng23 + Math.PI) % (2 * Math.PI) - Math.PI; //  
angle 1-2-3
```

```
if (Math.Sin(alpha1) == 0 && Math.Sin(alpha2) == 0) label3.Text = "Végtelen  
metszéspon";
```

```
if (Math.Sin(alpha1) * Math.Sin(alpha2) < 0) label3.Text = "Rossz adatok";
```

```
//alpha1 = Math.abs(alpha1);
```

```
//alpha2 = Math.abs(alpha2);
```

```
// ... Ed Williams takes abs of alpha1/alpha2, but seems to break calculation?
```

```
double alpha3 = Math.Acos(-Math.Cos(alpha1) * Math.Cos(alpha2) +  
Math.Sin(alpha1) * Math.Sin(alpha2) * Math.Cos(dist12));
```

```
double dist13 = Math.Atan2(Math.Sin(dist12) * Math.Sin(alpha1) *  
Math.Sin(alpha2), Math.Cos(alpha2) + Math.Cos(alpha1) * Math.Cos(alpha3));
```

```
double lat3r = Math.Asin(Math.Sin(lat1r) * Math.Cos(dist13) + Math.Cos(lat1r)  
* Math.Sin(dist13) * Math.Cos(brng13));
```

```
double dLon13 = Math.Atan2(Math.Sin(brng13) * Math.Sin(dist13) *  
Math.Cos(lat1r), Math.Cos(dist13) - Math.Sin(lat1r) * Math.Sin(lat3r));
```

```
double lon3r = lon1r + dLon13;
```

```
lon3r = (lon3r + 3 * Math.PI) % (2 * Math.PI) - Math.PI; // normalise to -  
180..+180°
```

```
lat3 = ToDegree(lat3r);
```

```
lon3 = ToDegree(lon3r);
```

```
lat3 = Math.Round(lat3, 5);
```

```
lon3 = Math.Round(lon3, 5);
```

```
}
```

```
private void button4_Click(object sender, EventArgs e)
```

```
{
```

```
    if (checkBox1.Checked == true) brng1 += 90;
```

```
    if (checkBox2.Checked == true) brng2 += 90;
```

```
    lat1 = Convert.ToDouble(TBlat1.Text);
```

```
    lat2 = Convert.ToDouble(TBlat2.Text);
```

```
    lon1 = Convert.ToDouble(TBlon1.Text);
```

```
    lon2 = Convert.ToDouble(TBlon2.Text);
```

```
    brng1 = Convert.ToDouble(TBdir1.Text);
```

```
    brng2 = Convert.ToDouble(TBdir2.Text);
```

```
    CalculateIntersection();
```

```
    TBlat3.Text = lat3.ToString();
```

```
    TBlon3.Text = lon3.ToString();
```

```
}
```

```
private void button5_Click(object sender, EventArgs e)
```

```
{
```

```
    TBdir1.Text = TBlat1.Text = TBlon1.Text = "";
```

```
}
```

```
private void button6_Click(object sender, EventArgs e)
```

```
{
```

```
    TBdir2.Text = TBlat2.Text = TBlon2.Text = "";
```

```
}
```

```
private void button7_Click(object sender, EventArgs e)
```

```
{
```

```
    lat3 = Convert.ToDouble(TBlat3.Text);
```

```
    lon3 = Convert.ToDouble(TBlon3.Text);
```

```

string url = "<script type='text/javascript'
src='http://www.google.com/jsapi'></script> <!DOCTYPE html><html><head><meta
name='viewport' content='initial-scale=1.0, user-scalable=no'> <meta charset='utf-8'><title>
</title> <style> html, body, #map-canvas { height: 100%; margin: 0px; padding: 0px }
</style><script
src='https://maps.googleapis.com/maps/api/js?key=AIzaSyD3pgdhZm_gSPIEIdd-
skZi19mtFABxvgA&sensor=false'> </script><script>function initialize() {var latA1 = " +
    lat3 +
    "; var lonA1 = " +
    lon3 +
    ";var PointA1 = new google.maps.LatLng(latA1,lonA1); var mapOptions =
{    zoom:10,    center:    PointA1    };var    map    =    new
google.maps.Map(document.getElementById('map-canvas'), mapOptions);var marker1 = new
google.maps.Marker(    {position:    PointA1,    map:    map,    title:    'Point1'}
);}google.maps.event.addDomListener(window, 'load', initialize);</script></head><body>
<div id='map-canvas'> </div> </body></html>";

```

```

using (FileStream fs = new FileStream("test.htm", FileMode.Create))
{
    using (StreamWriter w = new StreamWriter(fs, Encoding.UTF8))
    {
        w.WriteLine(url);
    }
}
Process.Start("test.htm");
}

}
}

```