

maXbox



maXbox Starter 34

Start with GPS

1.1 My Position please

Today we run through GPS coding.

Long day ago I did a lot of measures with my Garmin 310 device on the run and in trains; but because of security glasses, most of the trains disturb the signal. One of the questions that comes up when encoding those signals is how they can be useful in daily life.

Wiki says: The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the earth where there is an unobstructed line of sight to four or more GPS satellites.



👉 Each GPS satellite continuously broadcasts a navigation message at a rate of 50 bits per second. A message takes 750 seconds to complete! Such a record does have as minimum navi information:

procedure GPSRecord

begin

```
writeln('time,date,latitude,longitude,altitude,nsat,speed,course');
```

end

This isn't very different from a normal clock, isn't it? You'll notice that the altitude and speed takes part of the record. The speed is over ground in knots and the altitude means meters above sea level.

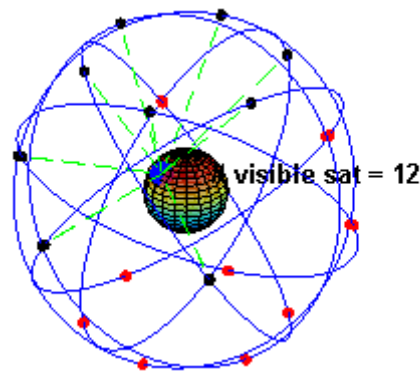
This information on NMEA sentences can be sourced from all over the net and NMEA is the National Marine Electronics Association. It is to be a worldwide, self-sustaining organization committed to enhancing the technology and safety of electronics used in marine applications and you see also a standard for GPS data formats.

But garmin like others have 6 proprietary sentences with 4 of them interpreted.

How it works: The GPS satellites transmit signals to a GPS receiver. These receivers passively receive those signals; they do not transmit and require a clear view of the sky, so they can only be used effectively outdoors, or from time to time in trains;-).

Type

```
TReceive_Func = TGPS.SerialRxChar(Sender: TObject);
```



Here's an interesting comparison. GPS signals are transmitted at a power equivalent to a 50 watt house light bulb. Those signal have to pass through space and our atmosphere before reaching your device after a journey of 11500 miles. Compare that with a TV signal, transmitt from a large tower 10 - 20 miles away at most, at a power level of 5-10000 watts.

All GPS satellites have atomic clocks. The signal that is sent out is a random sequence, each part of which is different from every other, called pseudo-random code (see pic below). This random sequence is repeated continuously. GPS receivers know this seq and repeat it internally.

So, satellites and the receivers must be in synch. The receiver picks up the satellite's transmission and compares the incoming signal to its own internal signal. In compare how much the satellite signal is lagging, the travel time becomes known.

1.1.1 Time to Code

Obviously the most important device required for this software to work, is the GPS receiver. All GPS devices with a NMEA0183 compatible connection are supported.

The GPS is used to get position, course, speed and information on NMEA quality and / or precision.

So I start with the call to the `Eye4Software` toolkit object:

```
begin //var objGps: variant;
  try
    objGps:= CreateOleObject('Eye4Software.Gps');
  except
    writeln('no GPS Obj: Invalid class string.');
```


Working with Free Pascal or Delphi you have to include some units to be able to use ActiveX controls in our project. We need to include the following units:

```
OleServer, ComObj, ActiveX.
```

In maXbox those units and many others are pre-compiled and included on demand. Now we are ready to declare the GPS constants as well.

```
objGpsConstants:= CreateOleObject('Eye4Software.GpsConstants');
```


After these steps, we can start with programming the rest of the GPS code. In order to connect a GPS receiver, you need to have at least one available serial COM port.

 If there is no port available, you can add a serial port by using an “USB-to-Serial” converter like on Arduino board available or a NMEA0183 data combiner equipped with an USB port.

```
ComboBoxDevice.Items.Add('Garmin USB');
for i:= 1 to 16 do begin
  ComboBoxDevice.Items.Add('COM'+ IntToStr(i));
end;
ComboBoxDevice.ItemIndex:= 1;
ComboBoxSpeed.Items.Add( '4800' );
ComboBoxSpeed.Items.Add( '9600' );
ComboBoxSpeed.ItemIndex:= 2;
```

When using such a converter, make sure it is connected at or before the time you are going to configure the GPS. I did also made some tests with `GarminAnt` agent direct calls, but that's another story.

When you have no control on which sentences are sent by the device, it is recommended to only select the specific GGA and VTG options in the NMEA0183 settings.

 The default serial baud rate for NMEA0183 devices is 4800bps. However, some hardware use other speeds (for instance, an AIS receiver will use 38400).

Now we discuss the start and stop procedure:


```
procedure ButtonStartClick(Sender: TObject);  
begin  
  objGps.DeviceSerialPort:= ComboBoxDevice.ItemIndex;  
  objGps.DeviceBaudrate:= StrToInt(ComboBoxSpeed.Text);  
  objGps.Open;  
  LabelStatus.Caption:= objGps.LastErrorDescription;  
  if (objGps.LastError = 0) then begin  
    ButtonStart.Enabled:= False;  
    ButtonStop.Enabled:= True;  
    Timer1.Enabled:= True;  
  end;  
end;
```

The interesting line is the timer which provides the incoming data:

```
procedure Timer1Timer(Sender: TObject);  
begin  
  EditLatitude.Text:= objGps.gpsLatitudeString;  
  EditLongitude.Text:= objGps.gpsLongitudeString;  
  EditSpeed.Text:= FloatToStr( objGps.gpsSpeed );  
  EditCourse.Text:= FloatToStr( objGps.gpsCourse );  
  EditAltitude.Text:= FloatToStr( objGps.gpsAltitude );  
  EditFix.Text:= IntToStr( objGps.gpsQuality );  
  EditSats.Text:= IntToStr( objGps.gpsSatellites );  
  EditTime.Text:= objGps.gpsTimeString;  
end;
```

Those data can be used with all GPS receiver that sends NMEA 0183 data, and as you know can be connected to a COM port on the computer.

This may however vary depending on what data sentence the connected GPS receiver delivers, for example some blow stuff:

 When the GPS receiver is set to change the DBR frequency or baud rate, the "J" sentence is replaced (just once) by (for example):
\$PSLIB,320.0,200*59 to set the DBR to 320 KHz, 200 baud.

The stop procedure is similar to open:

```
procedure ButtonStopClick(Sender: TObject);  
begin  
  objGps.Close;  
  LabelStatus.Caption:= objGps.LastErrorDescription;  
  if (objGps.LastError = 0) then begin  
    ButtonStart.Enabled:= True;  
    ButtonStop.Enabled:= False;  
    Timer1.Enabled:= False;  
  end;  
end;
```

So how do we picture those data? Most toolkits also include components that gives a graphical picture of satellite positions and signal strength.

We do it just simple from ASCII to a map tool (Map 60CSx):

Make your own tests with the whole track:

http://www.kleiner.ch/kleiner/garmin_paris_log.txt

Paris Trip (Strasbourg - Paris) with Garmin Map 60CSx

Geographische Koordinaten (dezimal) World Geodetic System 84 (WGS84)

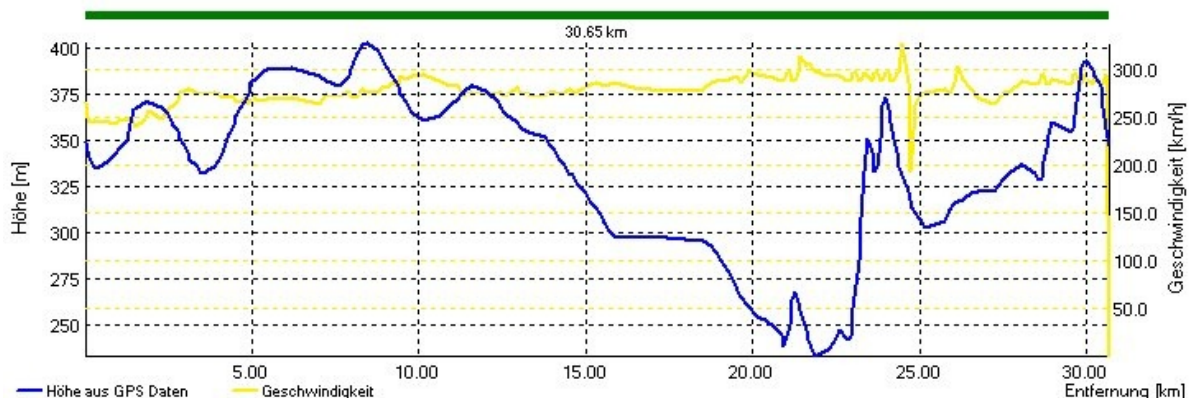
Nr.;Länge;Breite;Höhe [m];Zeit;Entf. [km];Kurs [°];Geschw. [km/h]

```
1;7.143274E;48.725447N;265.5;18.08.2009 09:48:01;0.000;0;0
2;7.140728E;48.725446N;266.9;18.08.2009 09:48:07;0.062;258;30
3;7.139905E;48.725334N;269.3;18.08.2009 09:48:09;0.310;266;31
4;7.135691E;48.725147N;270.3;18.08.2009 09:48:19;0.179;281;29
5;7.133298E;48.725440N;269.8;18.08.2009 09:48:25;0.127;301;31
6;7.131823E;48.726033N;269.8;18.08.2009 09:48:29;0.295;310;29
7;7.128757E;48.727752N;269.3;18.08.2009 09:48:39;0.031;322;31
```

In appendix you see the map and the coordinate in a red circle from which I started the recording. Those coordinates in the file are decimals and the longitude is before latitude, I would exchange that.

Variables, records or whole sentences can be assigned the value of the track to made a procedure that picture the trip in an image:

👉 Please note that the receiver must be set to transmitting NMEA data. The default setting for many receivers is a propriariety format!



You can also convert coordinates to a map on the internet (Appendix).

<http://www.gps-coordinates.net/>

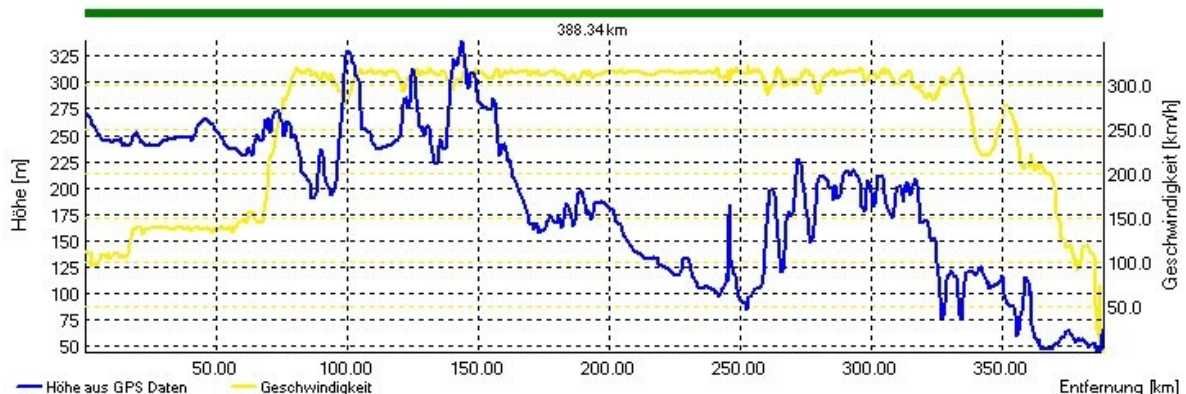
You can find the address corresponding to GPS coordinates or latitude, longitude and address of any point on Google Maps.

As more "things" on planet Earth are converted to the inventory set of digitally connected devices, the roles and responsibilities of web developers and technology leaders will need to evolve with GPS data.

Hope you did already work with the Starter 1 till 34 e.g. #15 and #19 with Serial and Arduino topics:

<http://sourceforge.net/p/maxbox/wikimax/main/>

Next we see the whole picture over a distance of 388 km. This image below is my sight of a track: the yellow line is the speed and the blue one marks the altitude point.

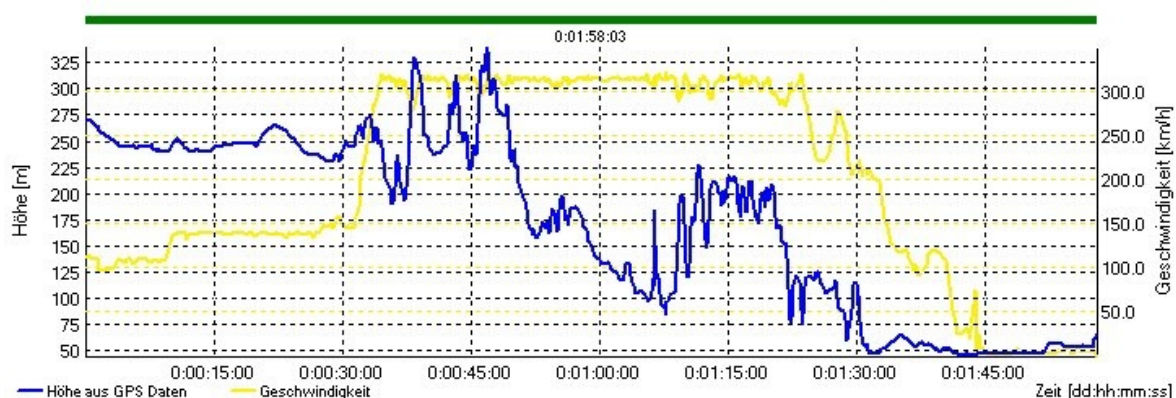


👉 Simplification: Your units are small and your methods too; you've said everything and you've removed the last piece of unnecessary code.

1.1.2 Get the Gold Code

I just said that concerning simplification to made a toast about the Gold code and his fascination. Because all of the satellite signals are modulated onto the **same** L1 carrier frequency, the signals must be separated after demodulation. This is done by assigning each satellite a unique binary sequence known as a Gold code. We can implement such a pseudo random routine with a closure.

Closures are reusable blocks of code that capture the environment and can be passed around as method arguments for immediate or deferred execution.



The **scope** of the GPS signal is missing as I show you now.

Gold sequences have been proposed by Gold in 1967 and 1968. These are constructed by EXOR-ing two m-sequences (ML) of the same length with each other.

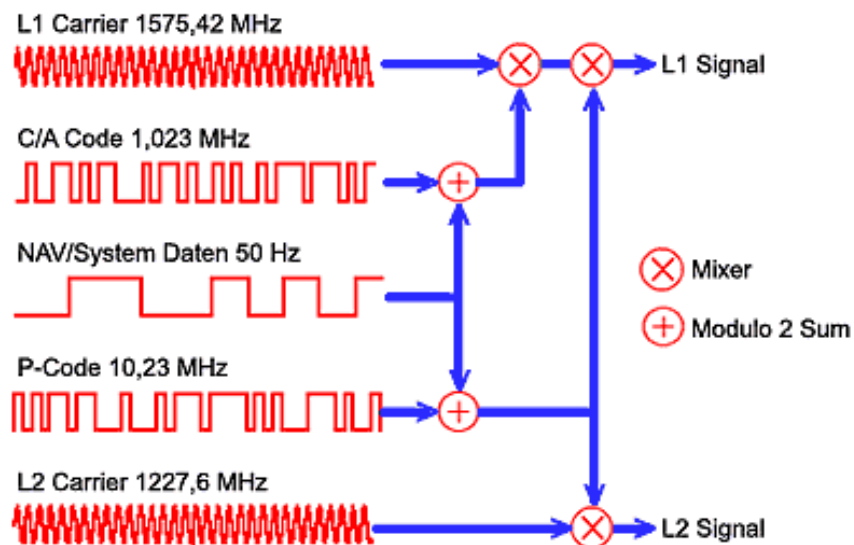
Maximal Length (ML or m) sequences or ML codes are well understood and have a number of properties which are useful in an application to spread-spectrum systems.

Gold codes have bounded small cross-correlations within a set, which is useful when multiple devices are broadcasting in the same frequency range like GPS with L1 signal.

A set of Gold codes can be generated with the following steps. Pick two maximum length sequences of the same length $2n - 1$ such that a cross-correlation is minimal:

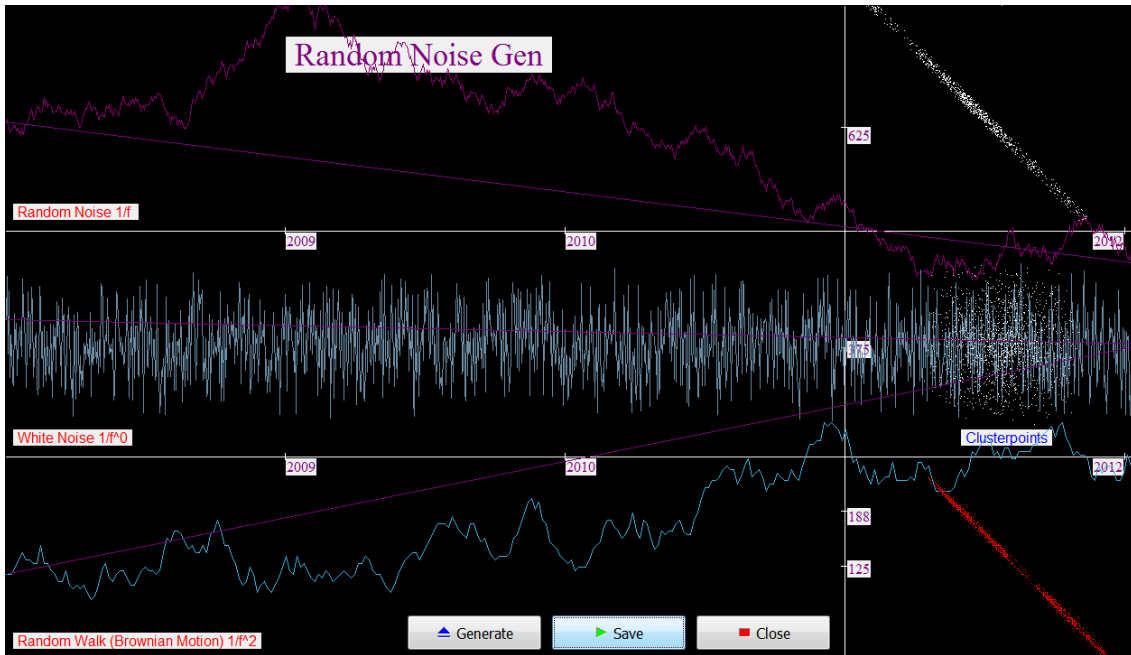
```
>>> refpow_to_N1 = generate_2^N-1 //maximum length
>>> refpow_to_N2 = generate_2^N-1 //maximum length
>>>> MINCrossCorrelation( refpow_to_N1, refpow_to_N2, <2(n+2)/2)
>>> XOR_to_N1N2 = generate_XOR(refpow_to_N1 , refpow_to_N2)
>>> GoldSet(XOR_to_N1N2)
```

The image below declares this context with arguments which a caller can follow in another context. You see also a P-code ("precision code"), a part of the positioning system signal.



Modulo 2 Sum (addition) is the XOR. So now you're wondering why I just spent this time explaining complicated things when it works. Just to give you some fascination with a conclusion:

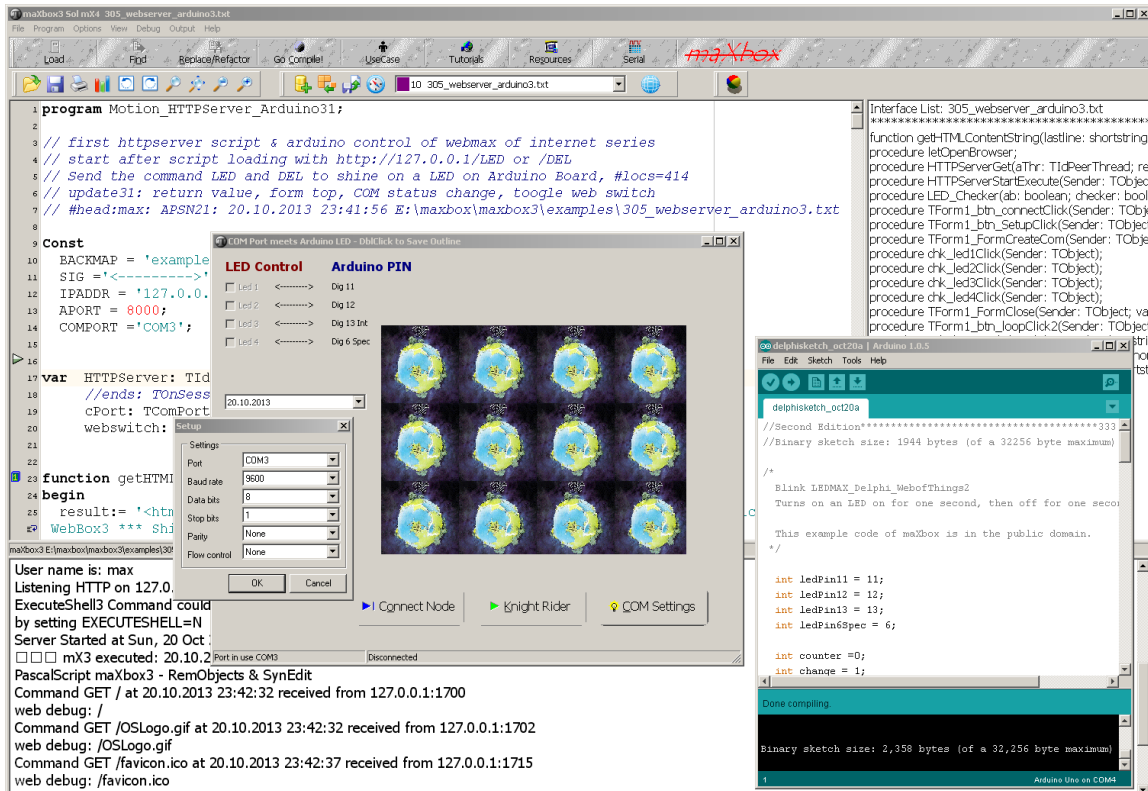
- In or demodulated signals must be separated by a Gold Code
- A unique binary sequence known as a Gold Code is assigned to each satellite
- The number attached to each signal bar identifies which satellites it's receiving a signal from.
- The pseudo random code is simply an I. D. code that identifies which satellite is transmitting information
- GPS satellites transmit two radio signals. These are designated as L1 and L2. Civilian GPS uses the L1 signal frequency (1575.42 MHz) in the UHF band.



7: The GUI of a Gold Code Generator

Almanac data is data that describes the orbital courses of the satellites. Every satellite will broadcast almanac data for EVERY satellite. Your GPS receiver uses this data to determine which satellites it expects to see in the local sky; then it can determine which satellites it should track.

👉 Almanac data is not precise and can be valid for many months.



8: COM Ports of a Serial Interface

maxbox

1.2 GPS and Arduino Conclusion

I would also recommend the book “Arduino Cookbook” by Michael Margolis. Here are a few ideas of more complicated projects that I have seen made with an Arduino.

- A box that will only open when it is at a certain location in the world (It connects a GPS to the Arduino. Search on “Reverse Geo-cache” to see some examples.)
- Or a controller for a 3D printer to print out the landscape the GPS has just scanned! (I'm just kidding)

Latency (sometimes called lag) is the time between a measurement has been made (for instance a position fix, or depth) and when the serial data is received by the application and Arduino.



Feedback @

max@kleiner.com

Literature:

Kleiner et al., Patterns konkret, 2003, Software & Support

<http://www.kleiner.ch/kleiner/gpsmax.htm>

http://www.softwareschule.ch/examples/475_GPS_mX2.txt

<http://www.pocketgpsworld.com/howgpsworks.php>

http://en.wikipedia.org/wiki/Global_Positioning_System#Message_format

- Software Manual of GPS Example.

<http://www.eye4software.com/files/hydromagic/manual.pdf>

http://www.softwareschule.ch/download/Arduino_C_2014_6_basta_box.pdf

<http://sourceforge.net/projects/maxbox>

1.3 Appendix Map Study



[maXmap](#)

DD (decimal degrees)*

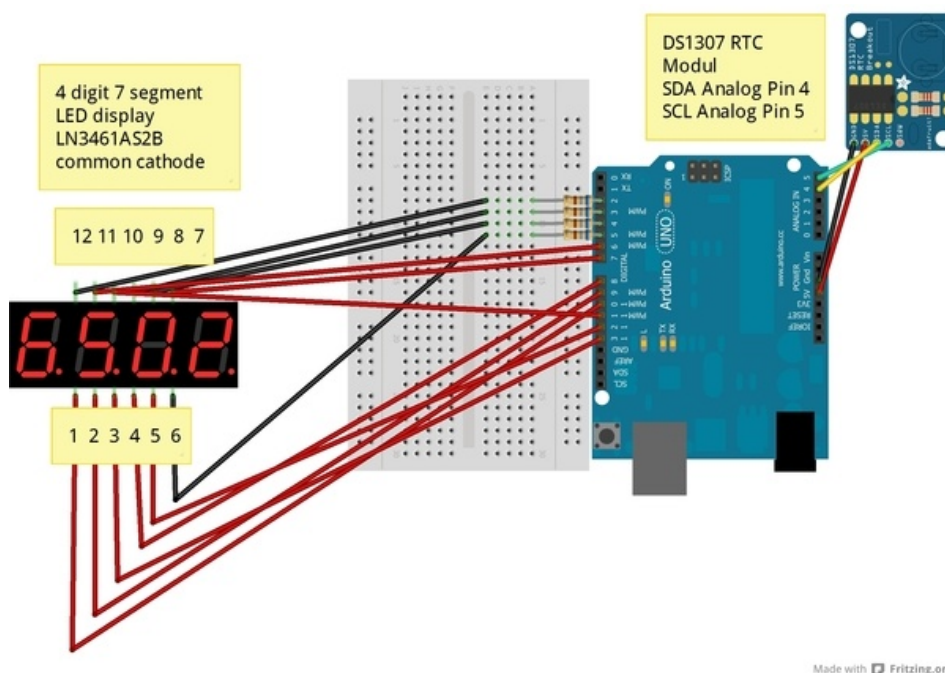
Latitude 48.725447N (48.656280963566495)

Longitude 7.143274E (6.991813648492098)

1 Rue du Canal, 57405 Guntzwiller, France

Latitude : 48.725447 | Longitude : 7.143274 Altitude : 287 meters

Here an embedded microcontroller system with a Real Time Clock:



RTClock: Arduino by Silvia Rothen <http://www.ecotronics.ch/ecotron/arduinocheatsheet.htm>