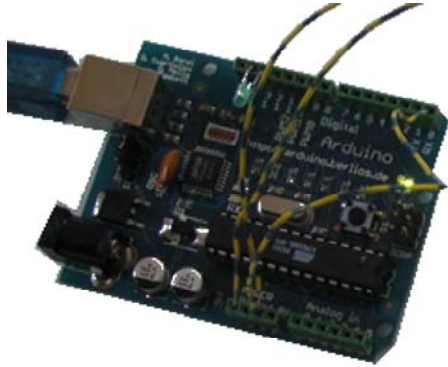




Comme son nom l'indique , ce multiplexeur NMEA permettra de recevoir des données de plusieurs sources différentes ( AIS, GPS, ....) et de les envoyer vers un PC , un iPad , un smartphone , une centrale de navigation ,....sur un port USB , un port Ethernet , un réseau wifi ,.....

Le cœur du système sera un contrôleur Arduino Méga 2560



Découverte du contrôleur Arduino

1 | La carte Arduino Méga 2560 et son câble USB



2 | Télécharger le logiciel Arduino

Dernière version sur la page [download page](#).

Déziper l' archive.....



*Remarque :il est possible d' utiliser une tablette Android au lieu d' un PC*



*L'application prend environ 210MB :assurez-vous que vous avez suffisamment d'espace libre dans la mémoire interne car elle ne peut être installée sur la carte SD à cause de la politique de sécurité d'Android.*

### 3 | Connecter la carte

Les cartes Arduino Méga reconnaissent automatiquement l' alimentation qu' elle vienne de la prise alimentation ( de 7 à 12 Volts ) ou de la prise USB

Connecter la carte au PC avec le câble USB. La LED verte s' allume .

### 4 | Installer les drivers

Dans le répertoire du logiciel Arduino aller dans le répertoire Drivers et double cliquer sur dpinst-amd64.exe si le PC Windows 7 ou 8 sont 64 bit sinon sur dpinst-x86.exe; sur mon PC de Nav avec XPSP3 le driver fourni avec le programme ne fonctionne pas , j' ai utilisé [celui ci](#).

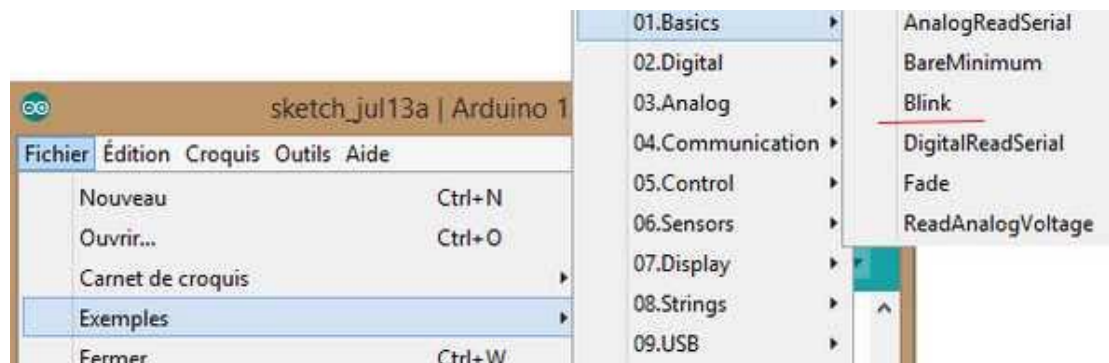
### 5 | Lancer l' application Arduino

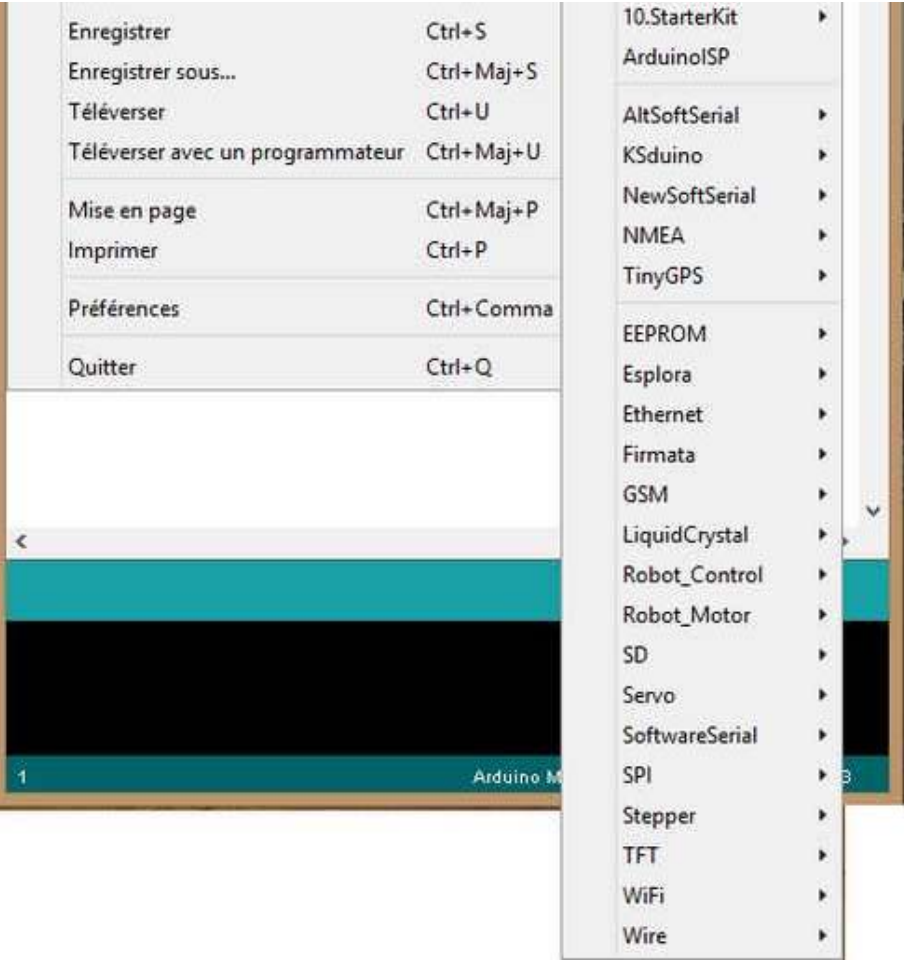


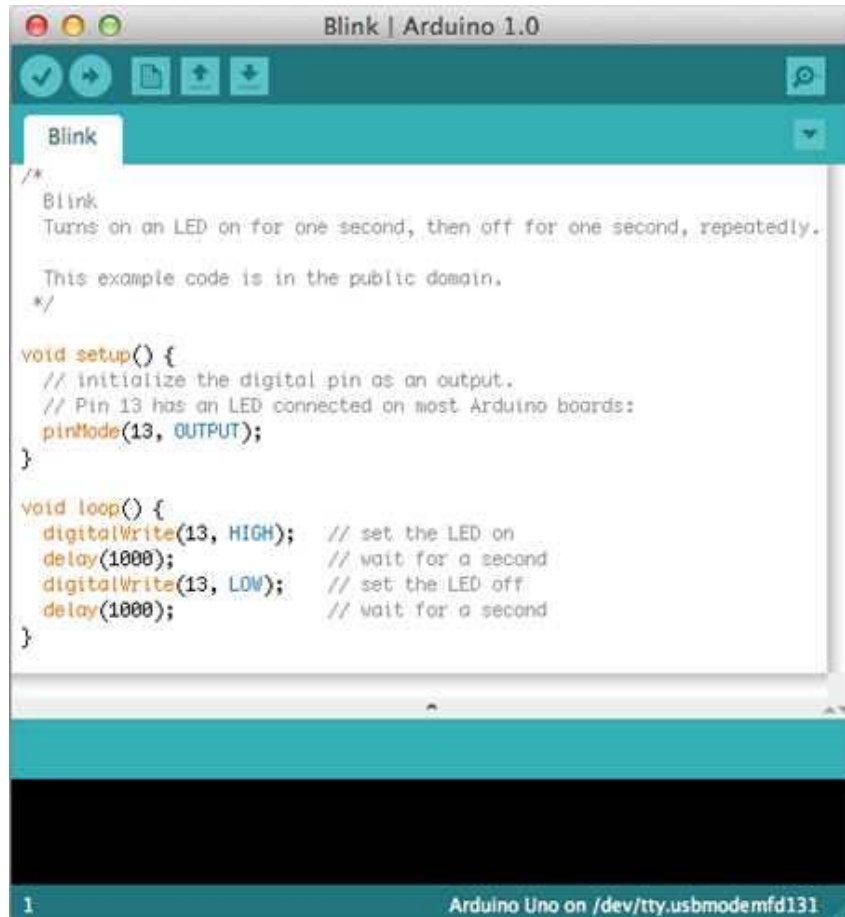
Le langage français doit être reconnu sinon le choisir dans FICHIERS/Préférences

#### 6 | Ouvrir l'exemple blink

Cet exemple permet de se familiariser avec le micro contrôleur: il s'agit de faire clignoter une LED à intervalle pré défini.





The image shows a screenshot of the Arduino IDE interface. The title bar at the top reads "Blink | Arduino 1.0". Below the title bar is a toolbar with icons for checking, uploading, saving, and other functions. The main text area contains the following code:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
  */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

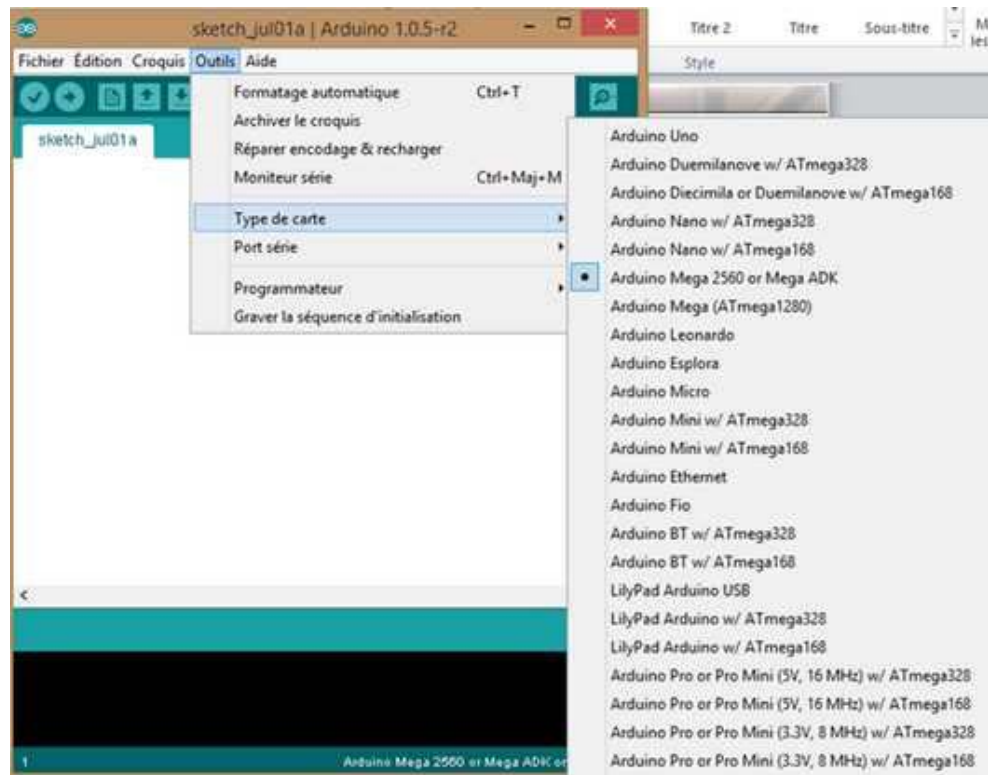
void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);             // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);             // wait for a second
}
```

The bottom status bar shows "1" on the left and "Arduino Uno on /dev/tty.usbmodemfd131" on the right.

Brancher une LED entre les Pin 13 et GND; une LED 12 volts de tableau électrique peut faire l'affaire ( elle éclairera faiblement).

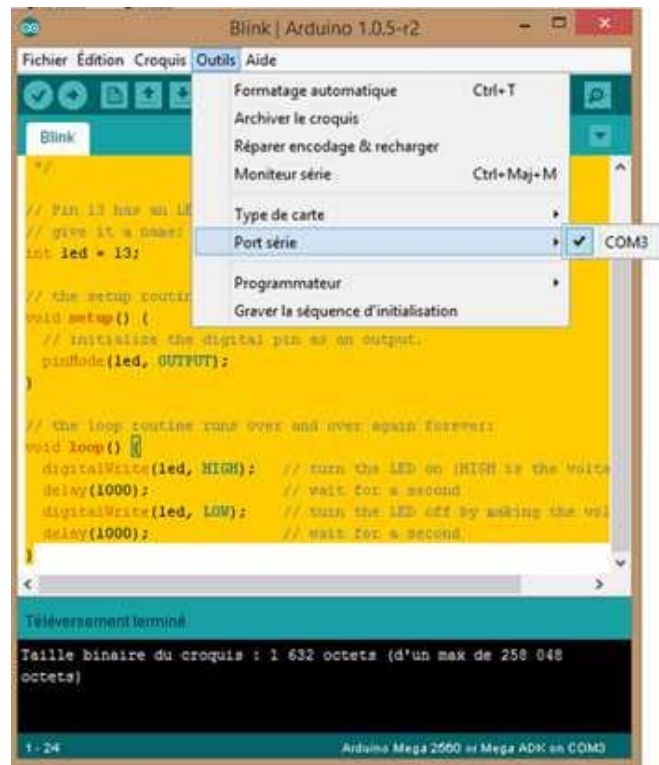


7 | Sélectionner le type de carte

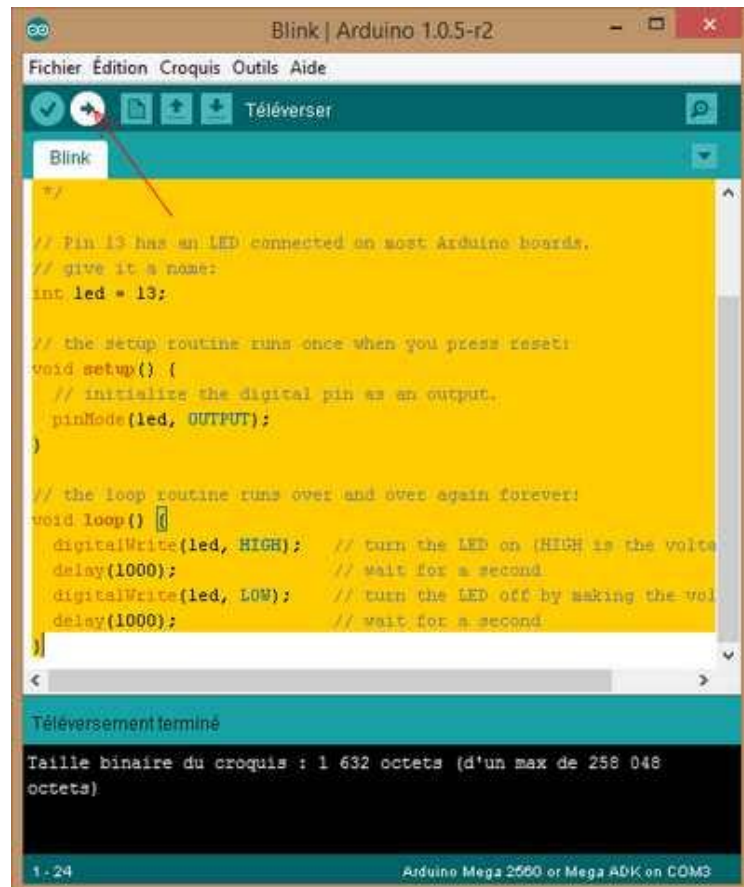


8 | Sélectionner le Port Série





## 9 | Uploader le Programme (téléverser)



Après quelques secondes , le programme est dans le micro contrôleur et démarre : la LED clignote  
Facile !!!!!.....les choses sérieuses maintenant:

Nous allons fabriquer un multiplexeur **NMEA 0183 USB et Wifi**, mais avec ce contrôleur il est possible de faire beaucoup plus ,.....un multiplexeur NMEA 0183 et 2000 , une centrale de navigation , etc ...

On peut lui adjoindre des modules complémentaires , Bluetooth , Wifi , GPS ,.... nous allons utiliser le module Ethernet.

Recevoir des entrées NMEA 0183 sur le contrôleur et les lire

L'interface électrique :

C'est l'interface série RS232 ,composée de 3 fils : Tx (émission), Rx (réception) et GND (référence) , qui est la plus utilisée par les équipements compatibles NMEA 0183(surtout le PC). L'interface différentielle RS422 ( composée de 4 fils (TX+ Tx-) et ( Rx+ Rx-) ) est aussi très utilisé utilisée pour la communication avec les appareils autres que le PC( VHF,lecteur de cartes,...) ; c' est à partir de cette dernière interface que nous allons construire notre multiplexeur ; il est néanmoins très facile de remplacer cette interface par une interface RS232 .....si nécessaire; je dis si nécessaire car les entrées-sorties NMEA à 3 fils des appareils ne sont pas de vrais RS232 au niveau des tensions qui varient entre le -5 v et le + 5 V , plus proches du standard RS422.(voir le tableau ci dessous qui indique le niveau de tension pour ces 2 standards)



les tensions varient de +5 Volts (niveau 0) à -5 Volts (niveau 1) par rapport à la masse (GND), voir tableau ci dessous)



La tension de à 5 Volts est envoyée entre + et - ( niveau 0) ou entre - et + ( niveau 1)



La tension entre Rx + et Rx - variera entre + 5 Volts et - 5 Volts :IMPORTANT ,NE PAS RELIER LE RX- à la MASSE

*C' est ce montage qu'il faudra employer pour notre multiplexeur en présence d' une liaison 3 fils*



C' est ce schéma par exemple de la sortie série 38400 bauds des VHF AIS Navicom ou Radio Océan; les fils sont bleu et gris plus la masse , entre bleu et gris c' est une liaison RS422 et entre bleu et la masse une liaison RS232.

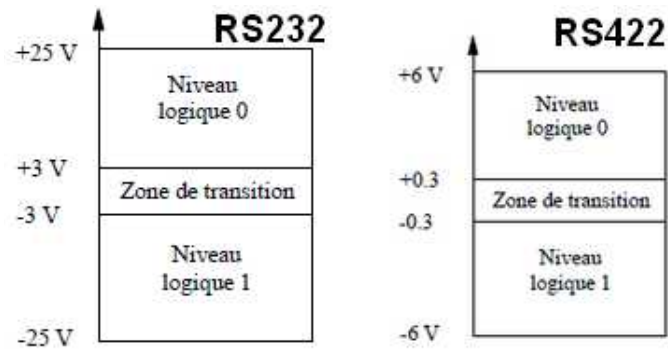
### Matériel nécessaire:

En plus de l' Arduino sur lequel les pins TX0 1 (transmit) et RX0 0 (receive) sont connectés à l' USB (donc le PC) et ne sont donc pas disponibles pour notre multiplexage. Les phrases NMEA reçu seront renvoyées sur ce port USB.

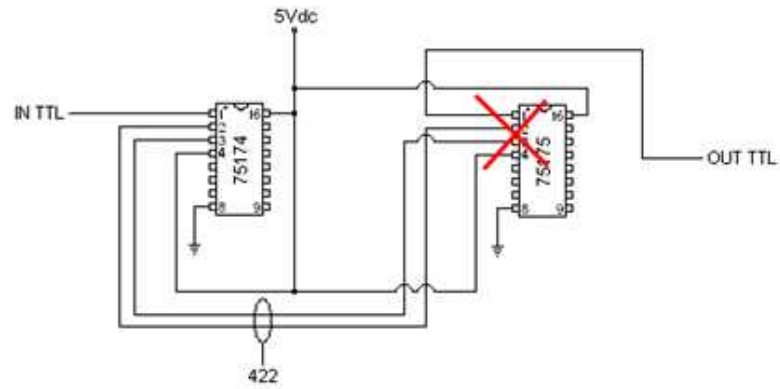
- Une platine d'essai et des fils de câblage pour la platine.
- Des CI 75175 et 75174 (ce dernier n'est utile que pour une sortie nmea , pas obligatoirement utile) pour convertir les signaux RS422 en TTL et réciproquement

Les micro contrôleurs n'utilisent pas de tensions négatives, ils utilisent des tensions de 0V et +5V; Un niveau logique 0 correspond à une tension de 0V et un niveau logique 1 correspond à une tension de +5V.

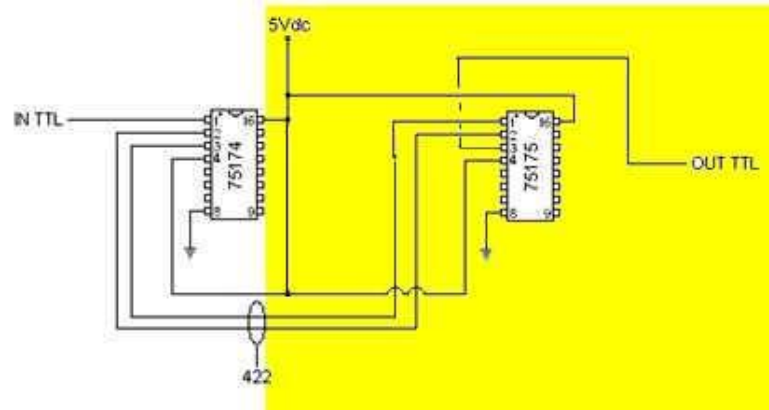
il faut donc adapter les tensions provenant des appareils car les normes RS232 et RS422 imposent les tensions suivantes :



Le montage souvent affiché sur Google est faux :

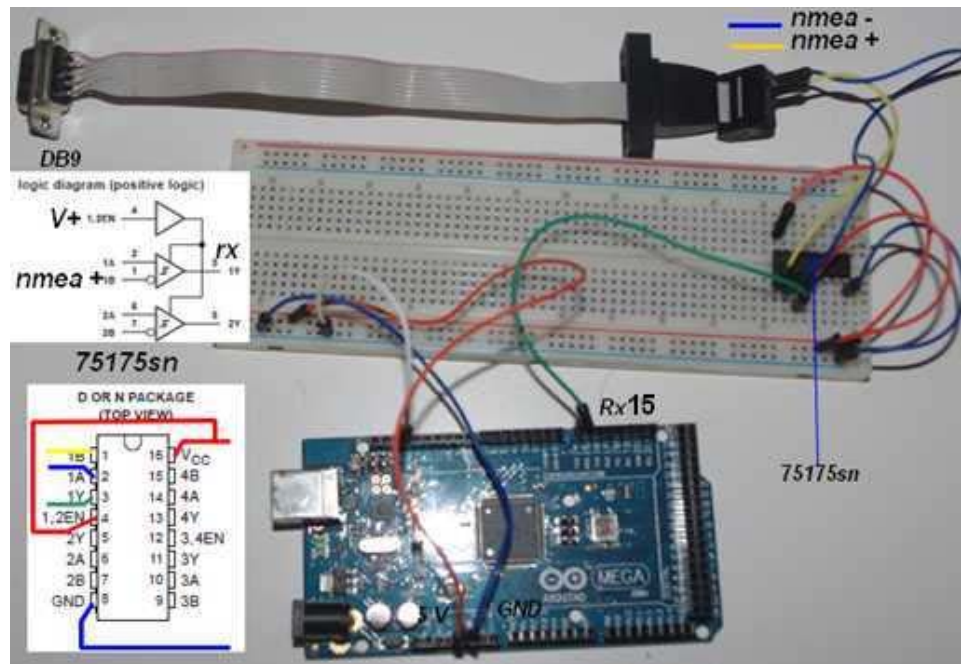


- 
- Le bon schéma :



- 
- Seule la partie en jaune nous intéresse pour l' instant.
- 

**Premier montage d'essai :**



Comme l'expliquent les schémas plus haut le fil NMEA 0183 - peut être remplacé par la masse(GND) si la liaison est 3 fils et ne provient pas d'un ordinateur (vraie RS232)

### ***Communication Série de la carte Arduino :***

*Port Série Serial : 0 (RX) et 1 (TX);*

*Port Série Serial 1: 19 (RX) et 18 (TX);*

*Port Série Serial 2: 17 (RX) et 16 (TX);*

*Port Série Serial 3: 15 (RX) et 14 (TX).*

*Ils sont utilisés pour recevoir (RX) et transmettre (TX) les données séries de niveau TTL. Les broches 0 (RX) et 1 (TX) sont connectées au circuit intégré ATmega8U2, le convertisseur USB-série de la carte.*

***Les broches d'alimentation sont les suivantes :***

*VIN. La tension d'entrée positive lorsque la carte Arduino est alimentée par une de tension externe . Il est possible d' alimenter la carte à l'aide de cette broche, ou, si l'alimentation se fait par le jack d'alimentation, utiliser cette tension en se raccordant sur cette broche.*

*5Volts. La tension régulée pour le micro contrôleur et les autres composants de la carte (un régulateur est intégré à la carte Arduino). .*

*3Volts3. Une alimentation de 3.3Volts pour certains circuits externes nécessitant cette tension à la place du 5Volts. L'intensité maximale disponible est de 50mA*

*GND. Broche de masse (0 Volt).*

**Le programme pour tester le montage :**

```
// -- programme de test lecture d' une entrée NMEA 0183
void setup()
{
  // Connexion série vers PV
  Serial.begin(9600); // 9600 = vitesse
  Serial.println("NMEA Multiplexer...Test Montage");
  // connexion NMEA .essai avec AIS de la VHF donc 38400
  Serial3.begin(38400) ; // 38400 = vitesse
}
void loop()
{char car = 0 ; //variable contenant le caractère a lire
int car_buffer = 0 ; //variable contenant le nb de caractères disponibles dans
le buffer
car_buffer = Serial3.available() ; // lecture nb caractères
```



```
// lecture caracteres
while(car_buffer > 0)
{
car = Serial3.read() ; //on lit le caractère
Serial.print(car) ; //envoi vers PC
car_buffer = Serial3.available() ; //lect a nouveau du nombre de caracteres
}
}
```

Les signaux provenant ou allant à la VHF sont utilisés ( la VHF est récente et ce sont de vrais signaux 422.

La vitesse d' envoi et réception est exprimée en bauds(bit par seconde)

Plus le câble est court, plus le débit pourra être élevé (moins de chute de tension et de parasites).

Le format des phrases NMEA :



AIS .....

**!AIVDM,1,1,,A,14eG;o@034o8sd<L9i:a;WF>062D,0\*7D**

!AIVDM: Le message NMEA type

1 Nombre de Sentences

1 Sentence Numéro

Pour messages multi-sentences)

A AIS Channel (A or B)

14eG;... Données AIS

0\* Fin des données

7D NMEA Checksum

GPS....

**\$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,\*47**

\$GPGGA : Type de trame

064036.289 : Trame envoyée à 06h40m36,289s (heure UTC)

4836.5375,N : Latitude 48,608958° Nord = 48°36'32.25" Nord

00740.9373,E : Longitude 7,682288° Est = 7°40'56.238" Est

1 : Type de positionnement (le 1 est un positionnement GPS)

04 : Nombre de satellites utilisés pour calculer les coordonnées

3.2 : Précision horizontale ou HDOP (Horizontal dilution of precision)

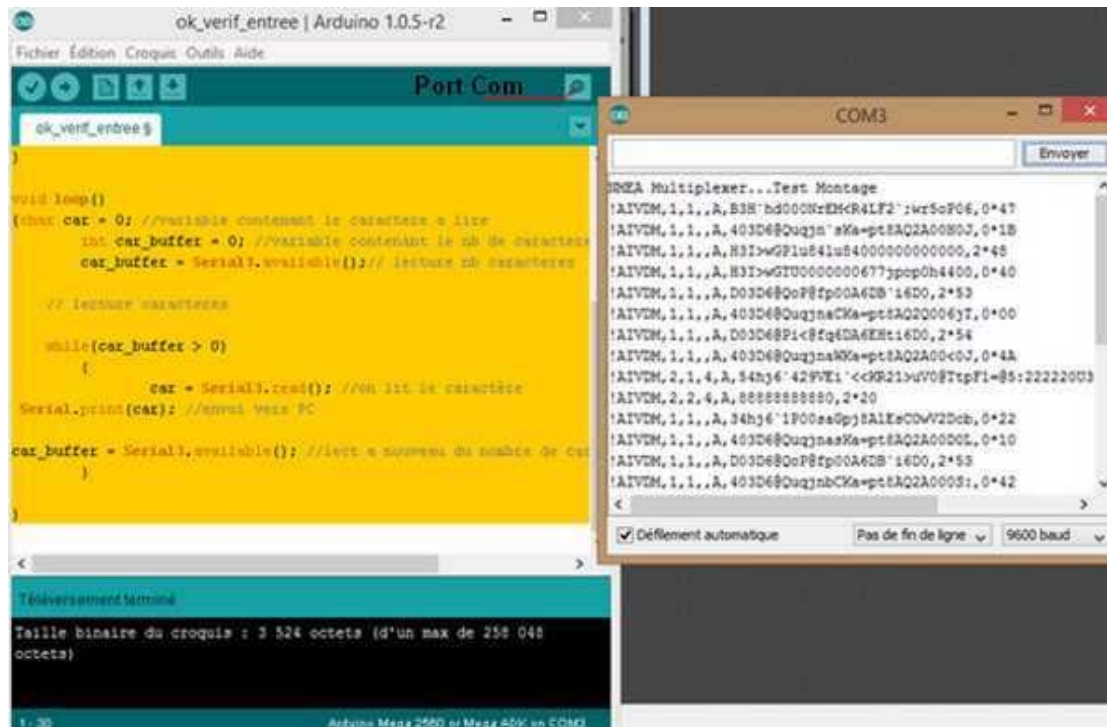
200.2,M : Altitude 200,2, en mètres

,,,,,0000 : D'autres informations peuvent être inscrites dans ces champs

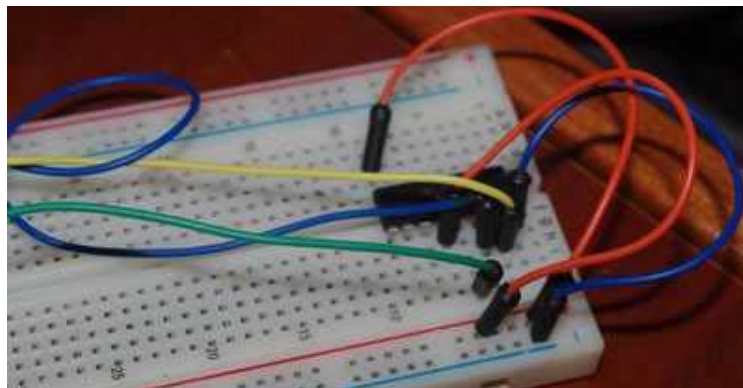
\*0E : Somme de contrôle de parité, un simple XOR sur les caractères précédents

Essai avec les signaux AIS à 38400 bauds .

Téléverser le programme dans l' Arduino et ouvrir le port Com .....Résultat :



Le 75175sn possède 4 entrées) : Essai sur la 2eme entrée avec le GPS à 4800 bauds



Remplacer les lignes du programme :

```
// connexion NMEA .essai avec AIS de la VHF donc 38400
```

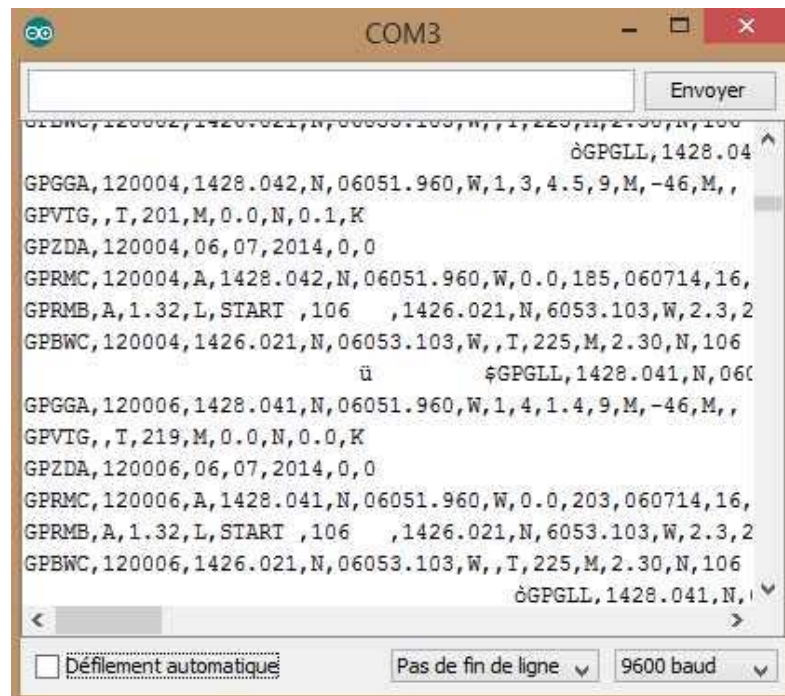
```
Serial3.begin(38400);
```

Par :

```
// connexion NMEA .essai avec le GPS a 4800
```

```
Serial3.begin(4800);
```

Ce qui donne :



Il faut maintenant multiplexer les données.

**Réalisation d' un multiplexeur 3 entrées NMEA0183 et 1 sortie USB**

Le programme pour le multiplexage :

il existe une librairie NMEA mais qui ne peut être utilisée , elle est uniquement consacrée au GPS , j' ai développé une librairie qui fait l' acquisition et le contrôle de validité des signaux NMEA provenant de différentes sources ( GPS, AIS , appareils propriétaires ,....):

Copier le contenu de chaque fichier dans un fichier texte et copier ces fichiers dans un répertoire nommé 'NMEA\_ACQ' ( 3 fichiers) ou téléchargez [NMEA\\_ACQ](#)

- Nmea\_acq.cpp
- 

```
/*
nmea_acq.cpp - Acquisition et vérification de validité de phrase NMEA
0183 $ (GPS...) et ! (AIS) pour Arduino
Copyright (c) 2014 Michel Gravier, France.
All right reserved.

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation;
This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301
USA
*/
#include "nmea_acq.h"
```

```
#define _LIB_version 1.01 // version de cette librairie
```

```
ACQ::ACQ(int connect)
{
    n = 0;
    _w = 0;
    _checksum = 0;
}

int ACQ::lecture(char c)
{
    // si LF et CR : _w=0 ----->case 0
    if ((c == 10) || (c == 13)) { _w = 0; }
    if ((c == '!') || (c == '$'))
    {
        _checksum = 0;
        _phrase[0] = c;
        n = 1;
        _w = 1;
        return 0; //valeur de retour FALSE
    }

    // ajout des caractères pour fabriquer la phrase
    switch(_w)
    {
        case 0:
            // attente des caractères '!' et '$'
            break;
        case 1:
            _phrase[n++] = c;
            switch (c)
            {
                case '*': // si fin de phrase ( avant checksum)
                    _w = 2; // il faut lire le 1er caractère du checksum en case 2
                    break;
                default: // ce n' est pas la fin de la phrase
                    _checksum = _checksum ^ c;
                    _w = 1;
                    break;
            }
            break;
        case 2:
            _phrase[n++] = c; // on lit le 1er caractère du checksum
```

```

    _check = (16 * _convert_hex(c));
    _w = 3; // il faut lire le 2eme caractère du checksum en case 3
    break;
    case 3:
        _phrase[n++] = c; // lecture 2eme caractere checksum
        _phrase[n++] = 13; // retour chariot
        _phrase[n++] = 10; // retour ligne
        _phrase[n++] = 0;
        _check = _check + _convert_hex(c);
        if (_checksum == _check) { _w = 0; return 1; } // phrase acceptée , valeur
        de retour TRUE pour 1
        else return 0; // valeur de retour FALSE pour 0
        default:

        break;
    }
    return 0;
}

```

```

char* ACQ::phrase() { // retourne la phrase valide
    return _phrase;
}

```

```

int ACQ::_convert_hex(char a) {
    // retourne la valeur en base 16 des caractères
    if (int(a) >= 65) { return int(a)-55; }
    else { return int(a)-48; }
}

```

```

int ACQ::libversion() {
    // retourne version de la librairie
    return _LIB_version;
}

```

- Nmea\_acq.h
-

```
/*  
nmea_acq.h - Acquisition et verification de validité de phrase NMEA 0183 $  
(GPS...) et ! (AIS) pour Arduino  
Copyright (c) 2014 Michel Gravier, France.  
All right reserved.
```

This library is free software; you can redistribute it and/or  
modify it under the terms of the GNU Lesser General Public  
License as published by the Free Software Foundation;  
This library is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
See the GNU  
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public  
License along with this library; if not, write to the Free Software  
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301  
USA

```
*/  
#ifndef nmea_acq_h  
#define nmea_acq_h  
  
#include "arduino.h"  
#define ALL  
0  
class ACQ  
{  
public:  
ACQ(int connect);  
int lecture(char c); // réception des caractères  
char* phrase(); // retour de la dernière phrase valide  
char _phrase[90]; // phrase nmea ( une phrase contient 82 caractères  
max)  
int n;  
int _w; // numero du traitement sur le caractere  
int _checksum; // somme de controle calculée  
int _check; // checksum inclus avec la phrase
```



```
int      _convert_hex(char a); //conversion en base 16
int  libversion() ;// affichage version
};
```

```
#endif
```

- keywords.txt
- 

```
# Datatypes (KEYWORD1)
#####
```

```
ACQ      KEYWORD1
```

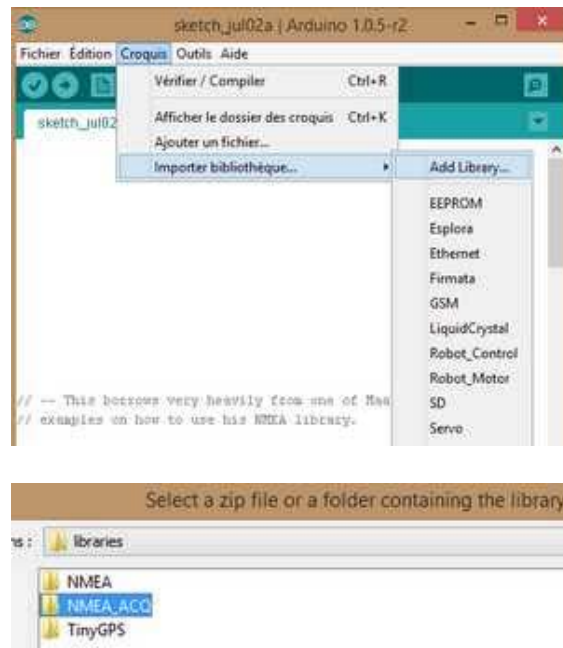
```
#####
# Methods and Functions (KEYWORD2)
#####
```

```
lecture KEYWORD2
phrase KEYWORD2
```

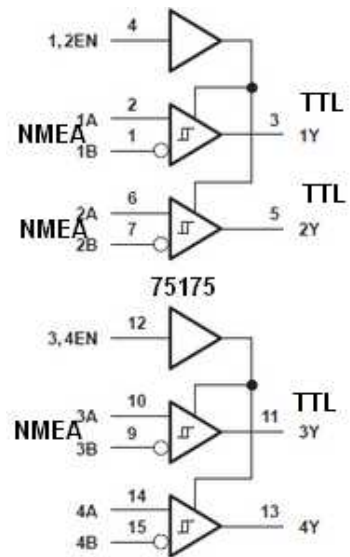
```
#####
# Constants (LITERAL1)
#####
```

```
ALL      LITERAL1
```

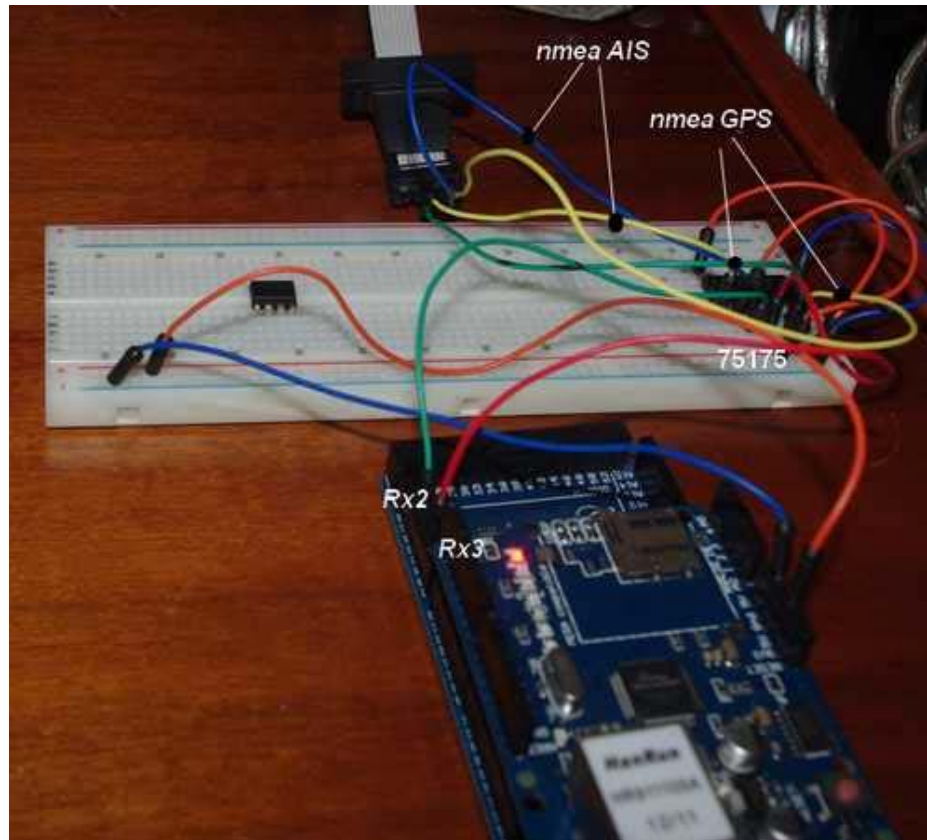
Importer la librairie :



Le CI 75175sn possède 4 entrées mais nous ne pouvons utiliser que 3 entrées de l' Arduino, car les E/S Rx0 et Tx0 ne peuvent être utilisées , elles sont utilisées par l' USB reliant le PC



l'essai se fera sur 2 entrées mais le programme fera l'acquisition de 3 entrées: rec\_gps , rec\_ais, rec\_nmeal



on définit les entrées par leur nom pour une meilleure lisibilité ( `#define .....` ), la partie réception des caractères est légèrement modifiée par rapport au programme précédent car elle utilise la librairie NMEA\_ACQ qui en plus de l'acquisition des caractères fera un contrôle de validité des phrases NMEA.

```
// Multiplexeur AIS + GPS + autres entrees  
// utilisation librairie NMEA_ACQ
```

```
#include <nmea_acq.h>//librairie nmea_acq
#define nmea_gps Serial3
#define nmea_ais Serial2
#define nmea_1 Serial1 // autre entree
//Serial ne peut être utilisé si le PC est connecté
ACQ rec_ais(ALL);// librairie pour AIS
ACQ rec_gps(ALL);
ACQ rec_nmea1(ALL);
;
void setup()
{
// Setup serial connexion vers PC.
Serial.begin(9600);
Serial.println("NMEA Multiplexer... GPS + AIS + Autres");

// Setup serial connexion NMEA
nmea_gps.begin(4800);
nmea_ais.begin(38400);
nmea_1.begin(4800); // autre entree
}

void loop()
{char car = 0; //variable contenant le caractère à lire GPS

// il y a un caractere à lire ?
if (nmea_gps.available())
{car = nmea_gps.read(); // lect caractere
// utilisation de la librairie NMEA_ACQ pour lire la sentence GPS
if (rec_gps.lecture(car)) // identique à : rec_gps.lecture(car)==1 , 1 pour
TRUE (vrai)
{// recuperation de la phrase nmea

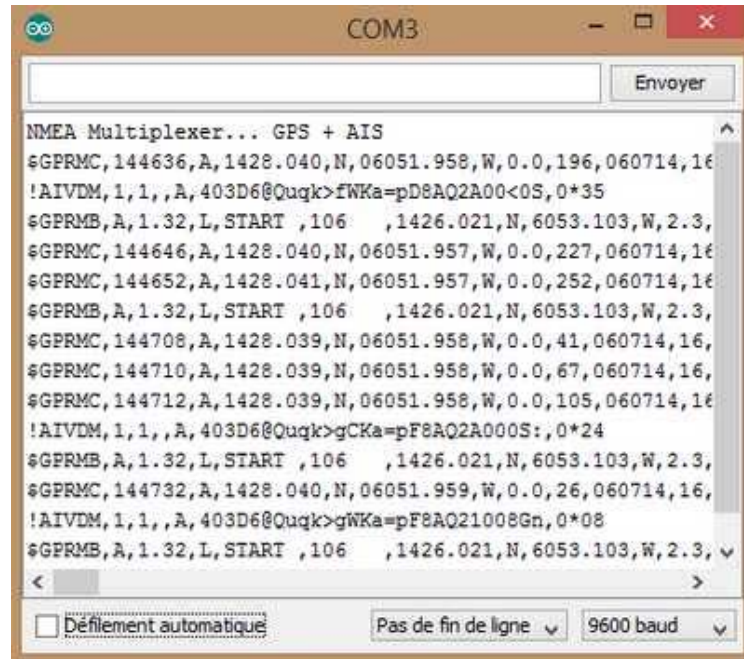
Serial.print(rec_gps.phrase());
}
}
if (nmea_ais.available())//
{car = nmea_ais.read(); // lect caractere
```

```
// utilisation de la librairie NMEA_ACQ pour lire la sentence AIS
if (rec_ais.lecture(car)) // si la condition est TRUE (vrai) , code retour
de la fonction
{ // recuperation de la phrase

Serial.print(rec_ais.phrase());
}
}
if (nmea_1.available())//
{car = nmea_1.read(); // lect caractere
// utilisation de la librairie NMEA_ACQ pour lire la sentence
if (rec_nmea1.lecture(car))
{ // recuperation de la phrase

Serial.print(rec_nmea1.phrase());
}
}
}
```

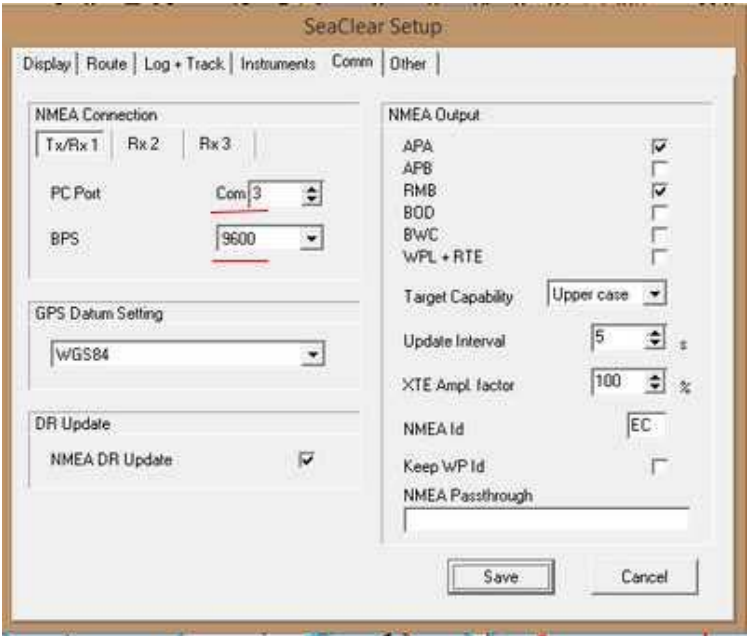
Résultat :



Il suffit maintenant , pour remplacer la plaque d'essai , de souder un support pour le CI sur une plaque de circuit imprimé pour gérer les signaux NMEA des différentes entrées afin de les renvoyer dans le format adapté (TTL) sur le contrôleur.

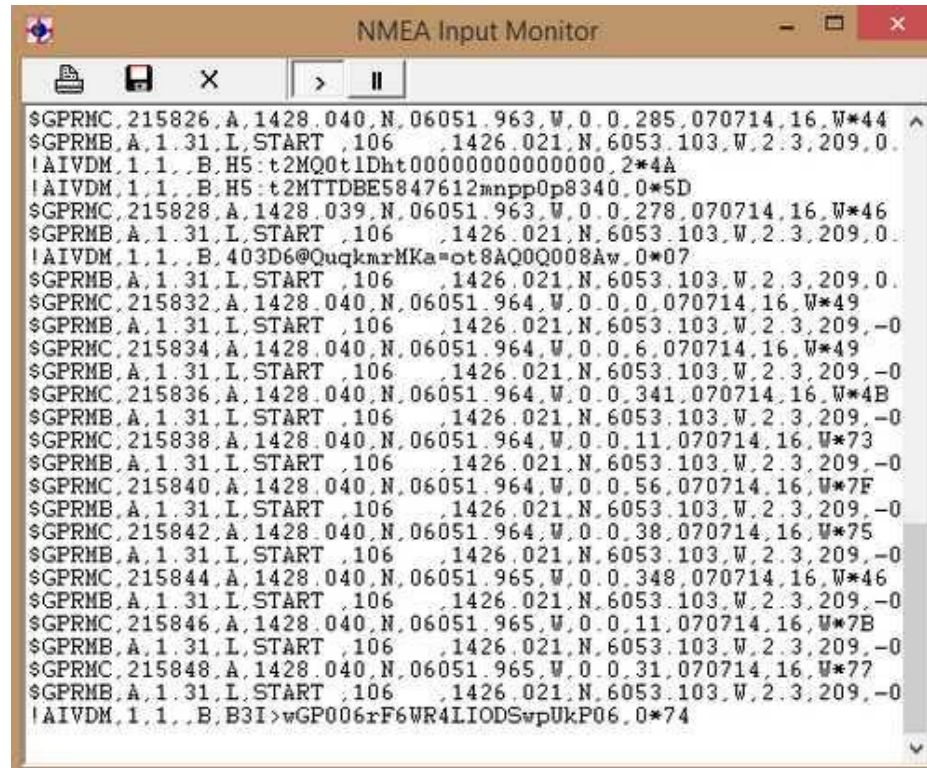
*Essai sous W7 et W8 , effectué avec SEACLEAR ( fermer le logiciel Arduino pour libérer le port com ).*

Utiliser le même port qu' Arduino.









```

NMEA Input Monitor
$GPRMC,215826,A,1428.040,N,06051.963,W,0.0,285.070714,16,W*44 ^
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,0.
!AIVDM,1,1,,B,H5:t2MQ0t1Dht00000000000000,2*4A
!AIVDM,1,1,,B,H5:t2MTTDBE5847612mnp0p8340,0*5D
$GPRMC,215828,A,1428.039,N,06051.963,W,0.0,278.070714,16,W*46
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,0.
!AIVDM,1,1,,B,403D6@QuqkMrMKa=ot8AQ0Q008Aw,0*07
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,0.
$GPRMC,215832,A,1428.040,N,06051.964,W,0.0,0.070714,16,W*49
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,-0
$GPRMC,215834,A,1428.040,N,06051.964,W,0.0,6.070714,16,W*49
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,-0
$GPRMC,215836,A,1428.040,N,06051.964,W,0.0,341.070714,16,W*4B
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,-0
$GPRMC,215838,A,1428.040,N,06051.964,W,0.0,11.070714,16,W*73
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,-0
$GPRMC,215840,A,1428.040,N,06051.964,W,0.0,56.070714,16,W*7F
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,-0
$GPRMC,215842,A,1428.040,N,06051.964,W,0.0,38.070714,16,W*75
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,-0
$GPRMC,215844,A,1428.040,N,06051.965,W,0.0,348.070714,16,W*46
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,-0
$GPRMC,215846,A,1428.040,N,06051.965,W,0.0,11.070714,16,W*7B
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,-0
$GPRMC,215848,A,1428.040,N,06051.965,W,0.0,31.070714,16,W*77
$GPRMB,A,1.31,L,START,106,1426.021,N,6053.103,W,2.3,209,-0
!AIVDM,1,1,,B,B3I>wGP006rF6WR4LIODSvpUKP06,0*74

```

Problème avec XP , le driver fourni avec l' application ne fonctionne pas correctement (essais avec avec OpenCPN et SeaPro) ; en installant ce driver le fonctionnement est correct avec SeaPro et OpenCPN ( version 3.2 )

Ce montage est un galop d'essai ,.....ci dessous ajout d' une sortie Ethernet

### Réalisation d' un multiplexeur 3 entrées NMEA 0183, 1 sortie USB et 1 sortie Ethernet

Nous allons simplement ajouter la communication Ethernet à notre montage .

Matériel complémentaire :

- une carte Arduino Ethernet ( elle possède également un lecteur de carte SD):

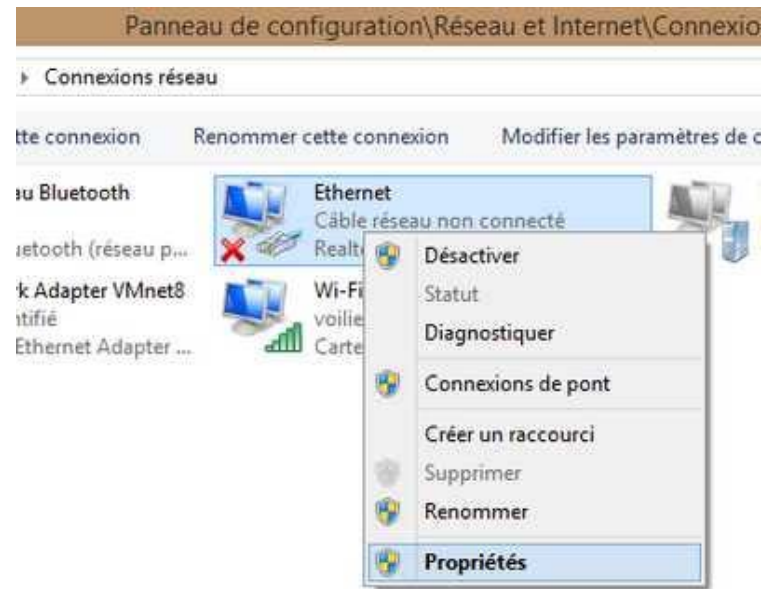


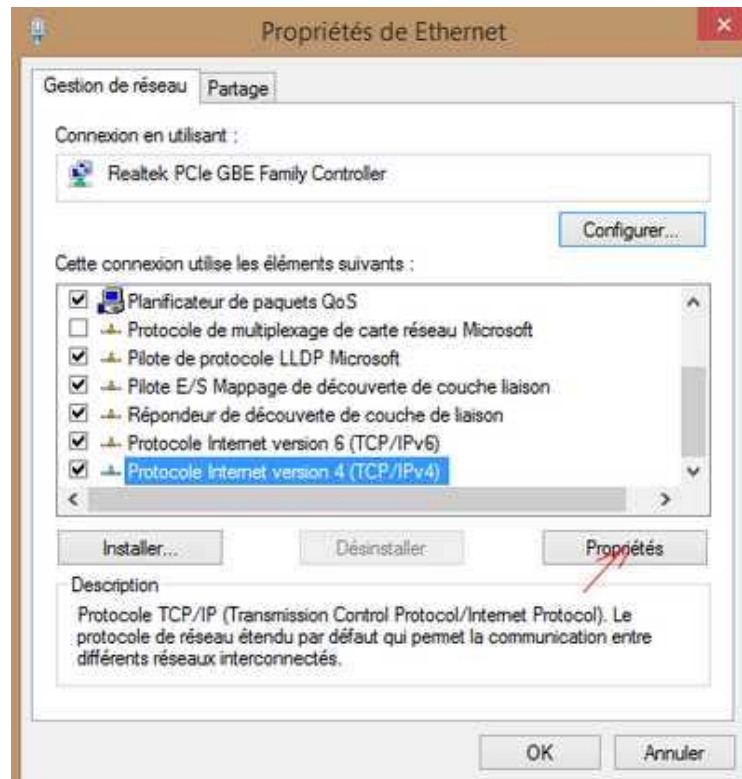
- il suffit de l' emboîter sur la carte principale :

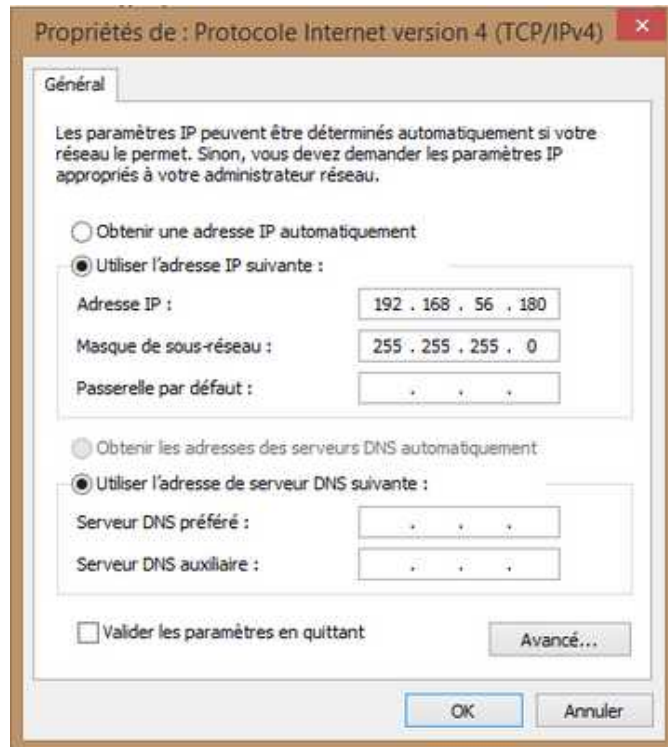


-

On relie avec un cordon Ethernet l'Arduino avec le PC; pour qu' ils puissent communiquer , il doivent être sur le même réseau , nous allons donc attribuer au PC l' adresse 192.168.56.180 ( l'arduino étant sur 192.168.56.177)







On complète le programme précédent :

```
// AIS + GPS + autres entrées + Ethernet
// utilisation librairie NMEA_ACQ

#include <nmea_acq.h> //librairie nmea_acq
#define nmea_gps Serial3
#define nmea_ais Serial2
#define nmea_1 Serial1 // autre entree
//Serial ne peut être utilisé si le PC est connecté
#include <SPI.h>
#include <Ethernet.h>
```



```
// (ethernet hardware) addresses de la carte (mac et IP):
byte mac = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip = { 192, 168, 56, 177 };
byte subnet = { 255, 255, 255, 0 };
EthernetServer server(80);

ACQ rec_ais(ALL); // librairie pour AIS
ACQ rec_gps(ALL);
ACQ rec_nmea1(ALL);

;
void setup()
{
  // initialialise la carte ethernet
  Ethernet.begin(mac, ip); // start ethernet using the mac and IP address
  server.begin();
  // Setup serial connexion vers PC.
  Serial.begin(9600);
  Serial.println("NMEA Multiplexer... GPS + AIS + Autres");

  // Setup serial connexion NMEA
  nmea_gps.begin(4800);
  nmea_ais.begin(38400);
  nmea_1.begin(4800); // autre entree
  delay(1000); // donne a la carte ethernet une seconde pour s' initialiser
}

void loop()
{
  char car = 0; //variable contenant le caractère à lire GPS

  // il y a un caractere à lire ?
  if (nmea_gps.available())
  {car = nmea_gps.read(); // lect caractere
  // utilisation de la librairie NMEA_ACQ pour lire la sentence GPS
  if (rec_gps.lecture(car)==1) // valeur de retour TRUE:1
```

```
{// recuperation de la phrase nmea
Serial.print(rec_gps.phrase());

    }

}
if (nmea_ais.available())//
{car = nmea_ais.read(); // lect caractere
// utilisation de la librairie NMEA_ACQ pour lire la sentence AIS
if (rec_ais.lecture(car)==1)// valeur retour TRUE:1
{// recuperation de la phrase

Serial.print(rec_ais.phrase());

}
}
if (nmea_1.available())//
{car = nmea_1.read(); // lect caractere
// utilisation de la librairie NMEA_ACQ pour lire la sentence
if (rec_nmea1.lecture(car))
{// recuperation de la phrase

Serial.print(rec_nmea1.phrase());
}
}
EthernetClient client = server.available();
if (client) {
boolean current_line_is_blank = true;
while (client.connected()) {
if (client.available()) {
char c = client.read();
// if we've gotten to the end of the line (received a newline
// character) and the line is blank, the http request has ended,
// so we can send a reply
if (c == 'n' && current_line_is_blank) {
// send a standard http response header
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
```



```
client.println();
// phrases NMEA à envoyer
client.println(rec_ais.phrase());
client.println(rec_gps.phrase());
break;
}
if (c == ' ') {
// we're starting a new line
current_line_is_blank = true;
} else if (c != ' ') {
// we've gotten a character on the current line
current_line_is_blank = false;
}
}
}
// give the web browser time to receive the data
delay(1);
client.stop();
}
}
```

Résultat sur un navigateur internet : il suffit de taper l' adresse indiquée dans le programme ( byte ip = { 192, 168, 56, 177 } ;)

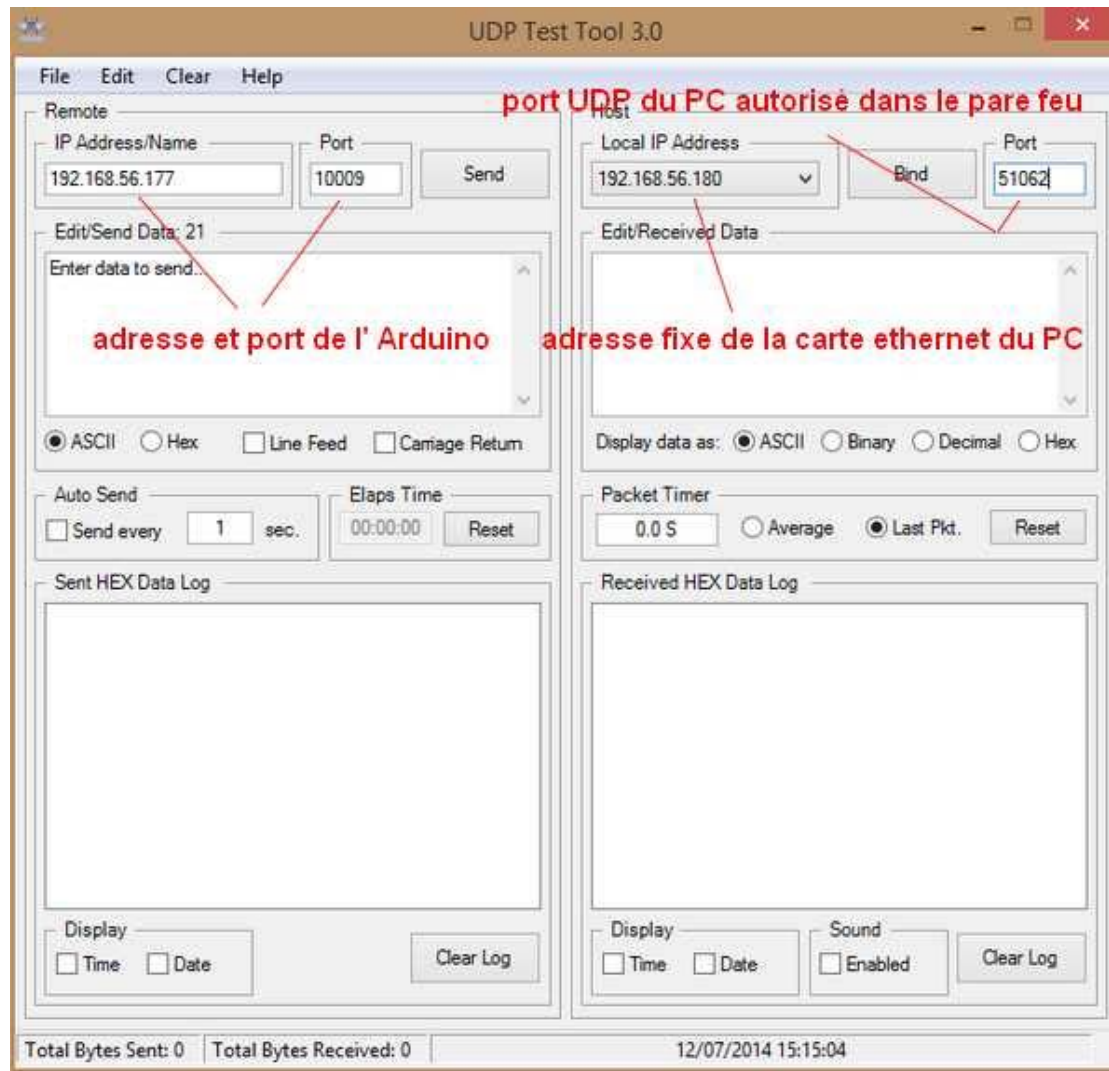


Nous voulons maintenant communiquer avec par exemple OpenCPN :

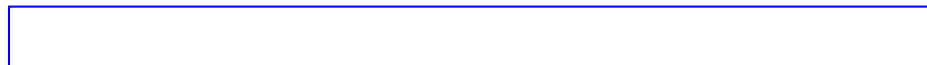
Il faut ajouter une bibliothèque qui gère les paquets : EthernetUdp.h ( pour le l' UDP); elle est comprise dans le programme Arduino , il suffit de la déclarer.

ATTENTION : il est impératif de créer une règle dans le pare feu pour autoriser les communications entrantes ; pour faire des essais on peut aussi désactiver le pare feu.

Un outil peut aider à régler les problèmes d' ouverture de ports UDP : UDP Test Tools



Le programme est à modifier comme ceci :



```
// AIS + GPS + autres entrées + Ethernet +UDP
// utilisation librairie NMEA_ACQ

#include <nmea_acq.h>//librairie nmea_acq
#define nmea_gps Serial3
#define nmea_ais Serial2
#define nmea_1 Serial1 // autre entree
//Serial ne peut être utilisé si le PC est connecté
#include <SPI.h>
#include <Ethernet.h>
//-----
#include <EthernetUdp.h>
// (ethernet hardware) addresses de la carte (mac et IP):
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 56, 177 };
IPAddress broadcastIP(192, 168, 56, 255);
byte subnet[] = { 255, 255, 255, 0 };
unsigned int localPort = 10009;
// -----
EthernetUDP Udp;
//-----
ACQ rec_ais(ALL);// librairie pour AIS
ACQ rec_gps(ALL);
ACQ rec_nmea1(ALL);

IPAddress ip_client(192, 168, 56, 180);// adresse du PC ou
// IPAddress ip_client(192, 168, 56,255);// adresse du réseau
unsigned int port_client=10110;// port UDP du PC ( 10110 port par default
d'OpenCPN)
char* buffer;
void setup()
{
// initialialise la carte ethernet
Ethernet.begin(mac, ip);
Udp.begin(localPort);
// Setup serial connexion vers PC.
Serial.begin(9600);
Serial.println("NMEA Multiplexer... GPS + AIS + Autres");
```

```
// Setup serial connexion NMEA
nmea_gps.begin(4800);
nmea_ais.begin(38400);
nmea_1.begin(4800); // autre entree
//delay(1000); // donne a la carte ethernet une seconde pour s' initialiser
}
void loop()

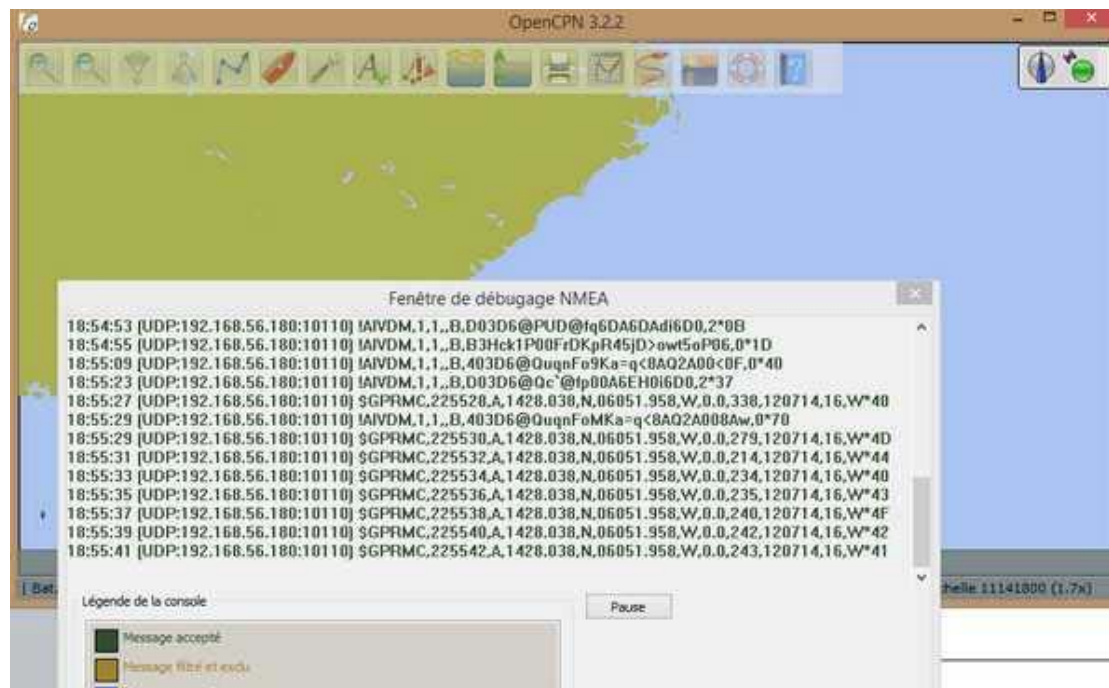
{
char car = 0; //variable contenant le caractère à lire GPS
// il y a un caractere à lire ?
if (nmea_gps.available())
{car = nmea_gps.read(); // lect caractere
// utilisation de la librairie NMEA_ACQ pour lire la sentence GPS
if (rec_gps.lecture(car))// valeur de retour TRUE
{// recuperation de la phrase nmea
Serial.print(rec_gps.phrase());
buffer=rec_gps.phrase(); // copy de la phrase nmea dans buffer
envoi_nmea(); // appel de la fonction d' envoi des paquets
}
}
}
if (nmea_ais.available())//
{car = nmea_ais.read(); // lect caractere
// utilisation de la librairie NMEA_ACQ pour lire la sentence AIS
if (rec_ais.lecture(car)==1)// valeur retour TRUE:1
{// recuperation de la phrase
Serial.print(rec_ais.phrase());
buffer=rec_ais.phrase();// copy de la phrase nmea dans buffer
envoi_nmea();// appel de la fonction d' envoi des paquets
}
}
}
if (nmea_1.available())//
{car = nmea_1.read(); // lect caractere
// utilisation de la librairie NMEA_ACQ pour lire la sentence
if (rec_nmea1.lecture(car))
{// recuperation de la phrase
Serial.print(rec_nmea1.phrase());
```

```

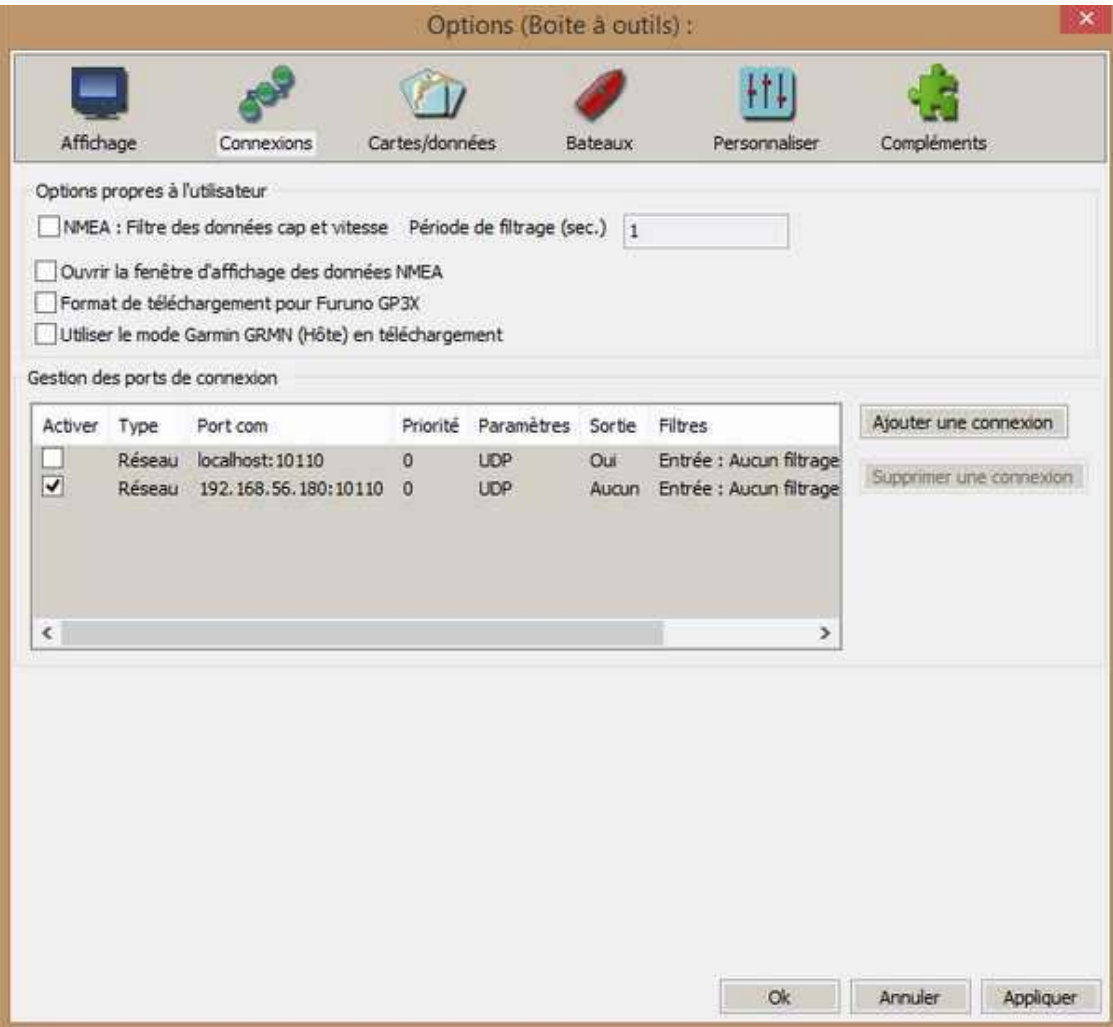
buffer=rec_nmea1.phrase();
envoi_nmea(); // appel de la fonction d'envoi des paquets }
}
}
void envoi_nmea() // fonction : envoi des paquets
{
Udp.beginPacket(ip_client, port_client);
Udp.write(buffer);
Udp.endPacket();
//
}

```

Communication avec OpenCPN établie par le câble Ethernet :



L' adresse ip du PC peut aussi être notée « Localhost » comme le montre l' image ci dessous ;



ci dessous : .....utilisation avec un routeur Wifi.

### Réalisation d' un multiplexeur NMEA 0183, Wifi (3 entrées , 2 sorties USB et Ethernet)

La carte wifi Arduino est chère alors qu'un routeur wifi est bon marché et peut être utilisé aussi en serveur multimédia et serveur de fichier.

Un routeur TP LINK ref MR3020 coûte entre 30 et 40 Euros ; il est compatible d-wrt , une petite merveille.



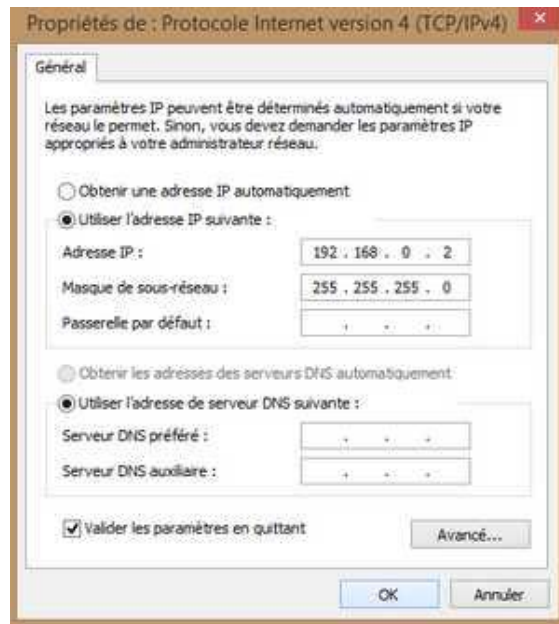
Le montage avec la Carte Arduino :





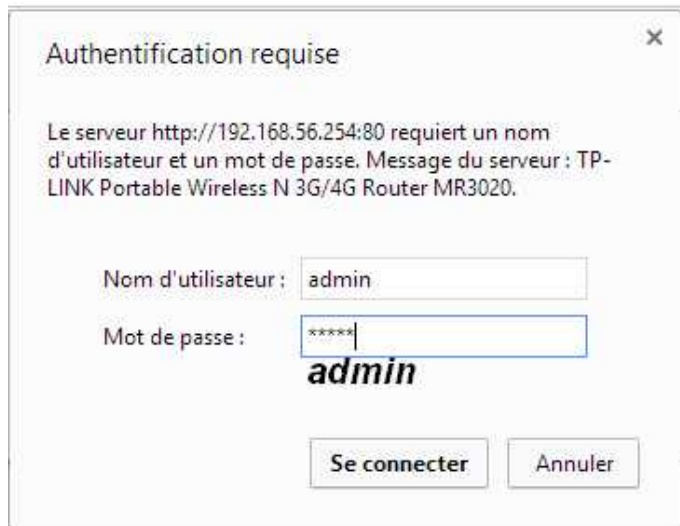
Il faut maintenant configurer le routeur , pour cela :

- comme pour les essais avec l' Arduino , le connecter avec le PC en Ethernet ( en ayant modifier l' adresse de la carte Ethernet pour être sur le même réseau que le routeur)



- avec un navigateur internet taper 192.168.0.254 ( sur certains routeur ce sera 192.168.0.1 , 192.168.1.1 ,... voir les notices constructeurs)

- renseigner le login et le mot de passe :



Authentication requise

Le serveur http://192.168.56.254:80 requiert un nom d'utilisateur et un mot de passe. Message du serveur : TP-LINK Portable Wireless N 3G/4G Router MR3020.

Nom d'utilisateur :

Mot de passe :   
**admin**

Si vous désirez aussi être connecté à internet par la wifi choisir Bridge avec AP sinon AP:



**TP-LINK**

Status  
Quick Setup  
WPS  
Network  
Wireless  
DHCP  
System Tools

**Quick Setup - Wireless Operation Mode**

Please Choose Operation Mode Type:

☐ Access Point (AP)  
☐ Repeater  
☒ Bridge with AP  
☐ Client

dans le cas de Bridge avec AP , choisir le réseau wifi pour l' accès à internet :

less Settings

Operation Mode: Bridge with AP

Wireless Network Name: vollier-idem

Region: France

Warning: Ensure you select a correct co  
Incorrect settings may cause it

Channel: 1

Mode: 11bgn mixed

Channel Width: Auto

☒ Enable Wireless Radio

☒ Enable SSID Broadcast

MAC of AP1: 00-3A-9A-13-11-B0

MAC of AP2:

MAC of AP3:

MAC of AP4:

Surviv

TL-MR3020 - Internet Explorer

http://192.168.56.254/overlappm/propupSiteSurveyRpm\_AP.htm?IMAC=mpk8Bsd42

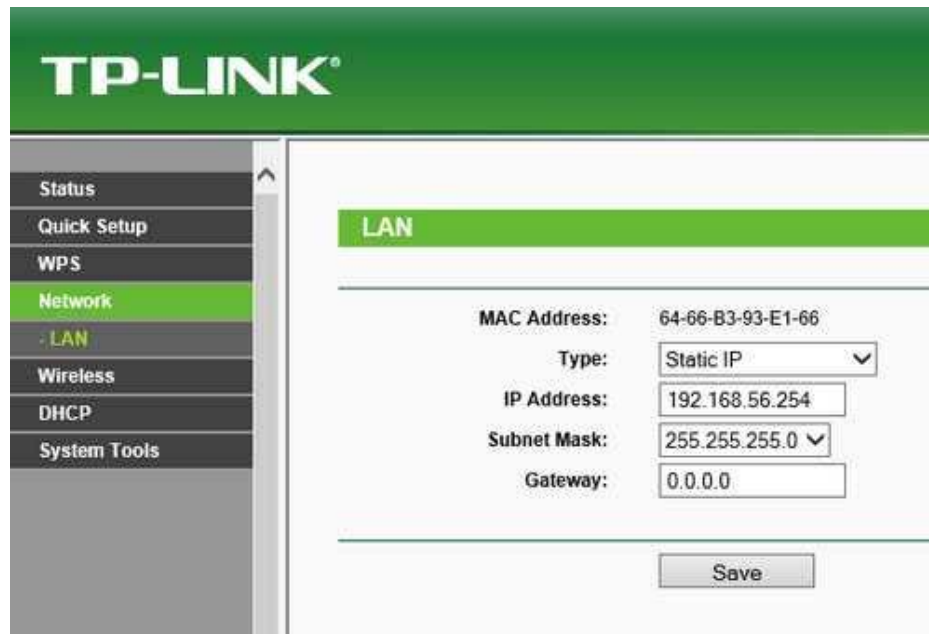
AP List

AP Count: 5

ID	BSSID	SSID	Signal	Channel	Security	Choose
1	DC-9F-DB-54-B2-2B	Bad Boy	9dB	1	ON	<a href="#">Connect</a>
2	00-3A-9A-13-11-B0	Marina_du_Marin	4dB	1	OFF	<a href="#">Connect</a>
3	00-27-22-08-AD-8A		17dB	3	ON	<a href="#">Connect</a>
4	00-3A-9A-13-13-60	Marina_du_Marin	18dB	5	OFF	<a href="#">Connect</a>
5	08-7A-4C-6D-9D-B7	Orange-9db7	21dB	11	ON	<a href="#">Connect</a>

Refresh

Choisir l' adresse du réseau : 192.168.xxx.254



The image shows the TP-LINK web interface for configuring network settings. The left sidebar contains a menu with the following items: Status, Quick Setup, WPS, Network (highlighted), LAN (highlighted), Wireless, DHCP, and System Tools. The main content area is titled 'LAN' and displays the following configuration fields:

MAC Address:	64-66-B3-93-E1-66
Type:	Static IP
IP Address:	192.168.56.254
Subnet Mask:	255.255.255.0
Gateway:	0.0.0.0

A 'Save' button is located at the bottom of the configuration area.

Choisir la plage d'attribution des adresses IP pour le réseau Wifi :

The screenshot shows the TP-LINK web interface for DHCP settings. The left sidebar contains a menu with the following items: Status, Quick Setup, WPS, Network, Wireless, DHCP (highlighted), DHCP Settings (highlighted), DHCP Clients List, Address Reservation, and System Tools. The main content area is titled 'DHCP Settings' and contains the following configuration options:

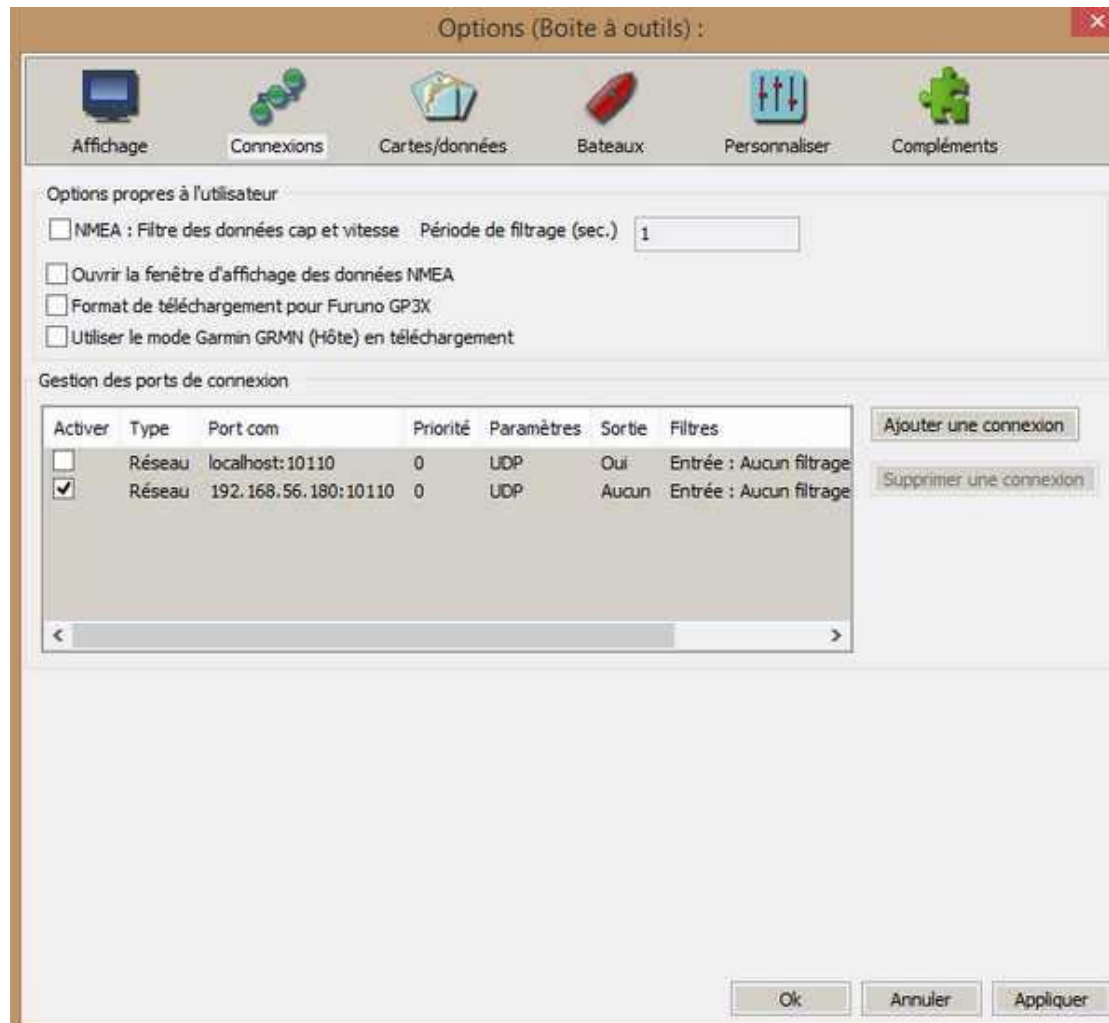
- DHCP Server: ☐ Disable ☒ Enable
- Start IP Address: 192.168.56.100
- End IP Address: 192.168.56.199
- Address Lease Time: 120 minutes (1~2580 minutes, the default value is 120)
- Default Gateway: 192.168.56.254 (optional)
- Default Domain: (optional)
- Primary DNS: 0.0.0.0 (optional)
- Secondary DNS: 0.0.0.0 (optional)

A 'Save' button is located at the bottom of the form.

D' autres paramètres sont à choisir comme par exemple la sécurité wifi ,.....

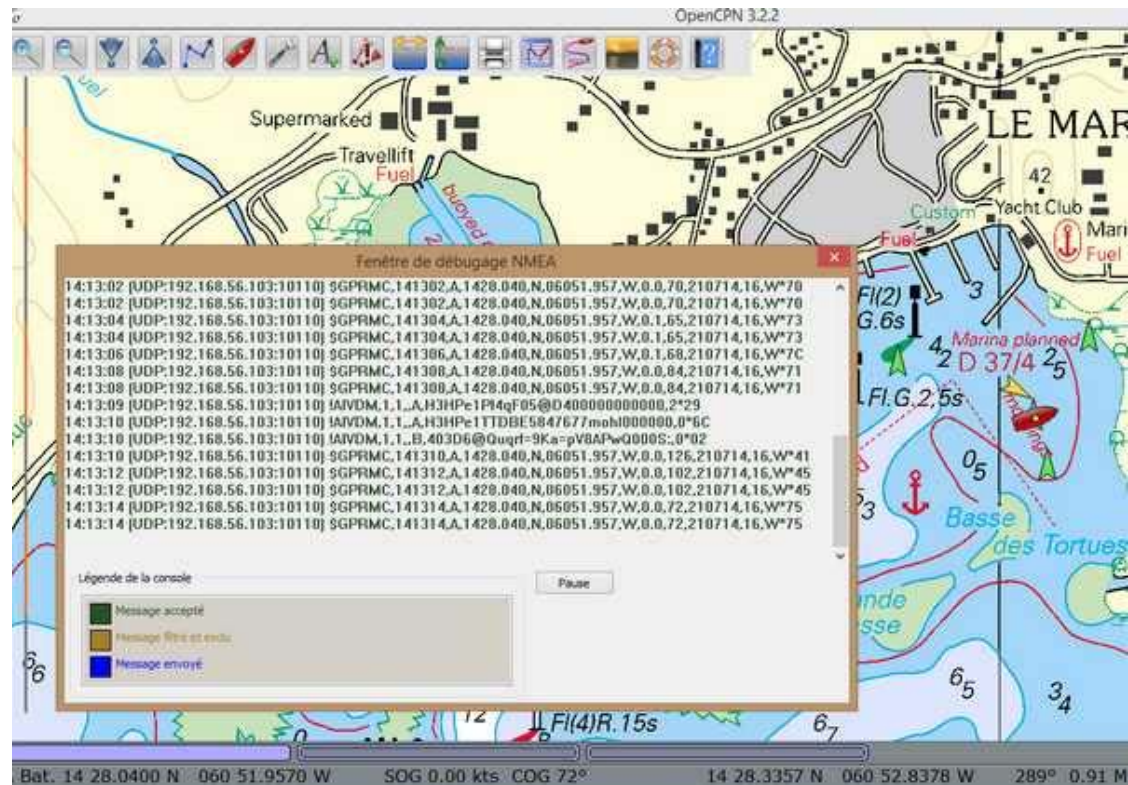
Il n' y a plus qu'à faire un essai :

- avec OpenCPN:



Je n' ai pas changé l' adresse , maintenant le PC utilise la carte wifi et pourtant cela fonctionne ,l' important c' est le port UDP car la carte Arduino envoie les données à tous les ordinateurs du réseau si cette option a été choisie : IPAddress ip\_client(192, 168, 56,255);// adresse du réseau

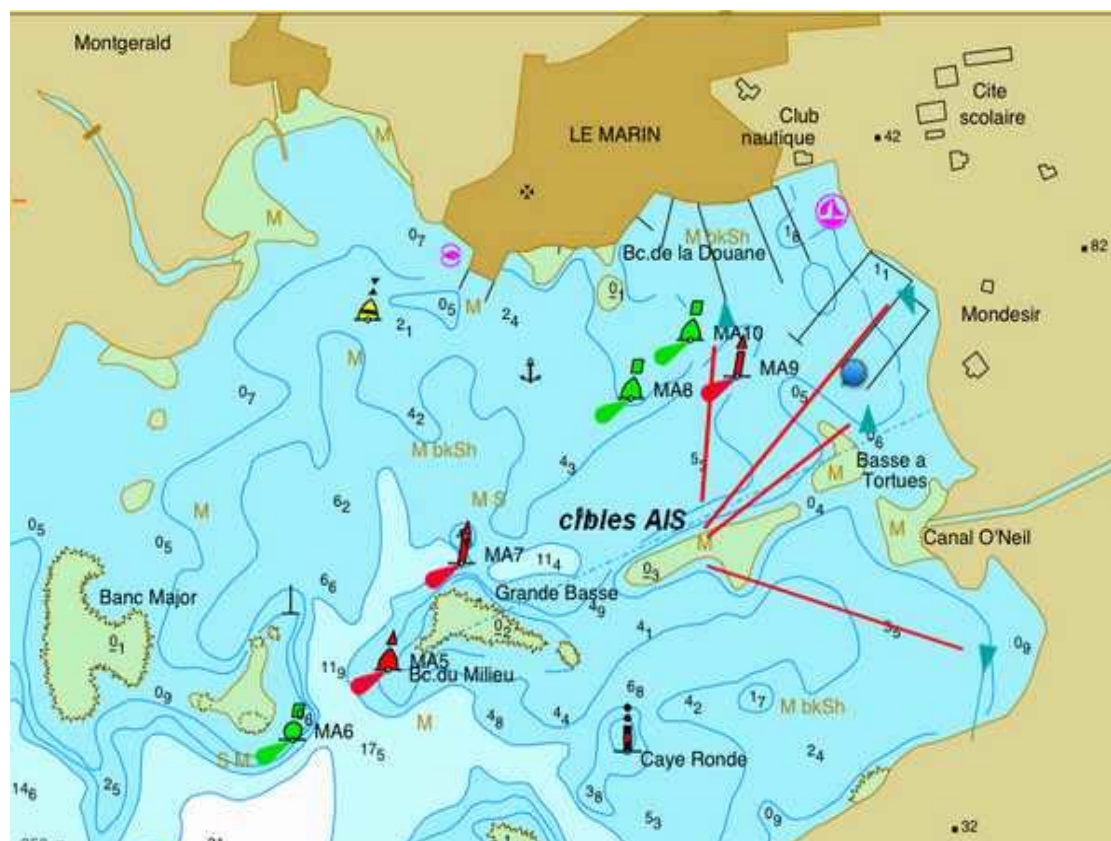
J' ai utilisé l' adresse de la carte wifi :



En résumé , avec OpenCPN , si on utilise comme adresse *localhost* les données sont toujours prises en compte; c' est le port UDP qui est important.

### Visualisation des données du multiplexeur sur l' iPad

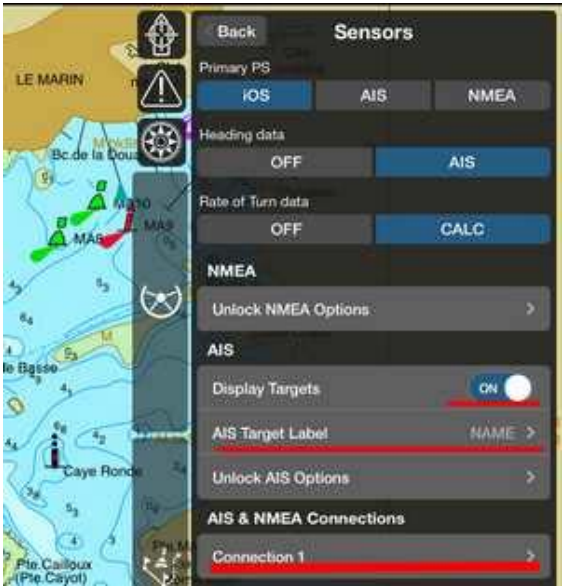




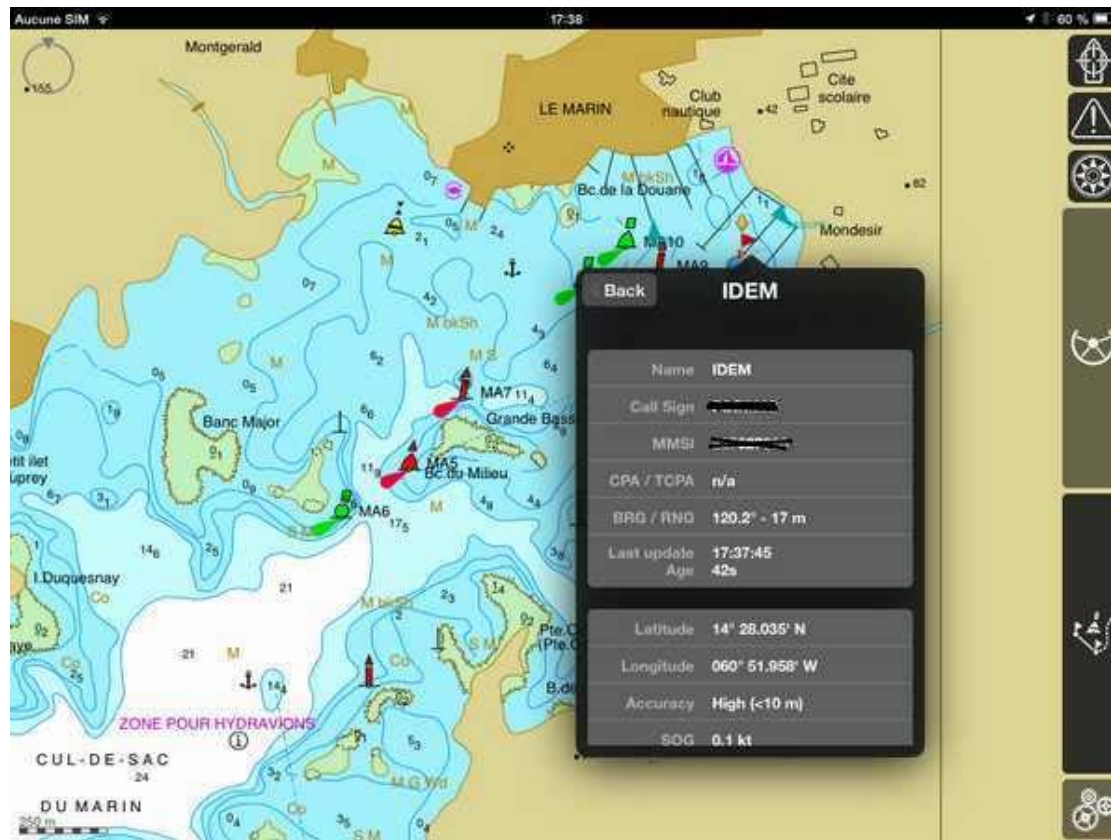
Eh oui ça fonctionne comme sur le sur PC , voici la marche à suivre :

Nous avons l' iPad , iSailor avec des cartes mais sans l' option AIS; Pour l' instant iSailor nous a couté 21,99 Euros , nous allons donc investir en achetant l' option AIS à 8,99 Euros.

Le déblocage de cette option s' effectue rapidement sans problème , il faut paramétrer la réception des données:



Voici les informations concernant Idem:



## CONCLUSION:

Ce montage fonctionne sur PC , sur iPad , il devrait fonctionner sur les tablettes Android , Windows 8.1 et Windows RT8.1.

créé le 13/07/2014