Montage pour enregistrer les données d'un port série sur une carte SD

Semble être le mieux documenté…

http://patricktiercelin.wixsite.com/logiciels-tiercelin1/data-logger

Ce montage est optimisé pour enregistrer les données du Livre de Bord (version 8)

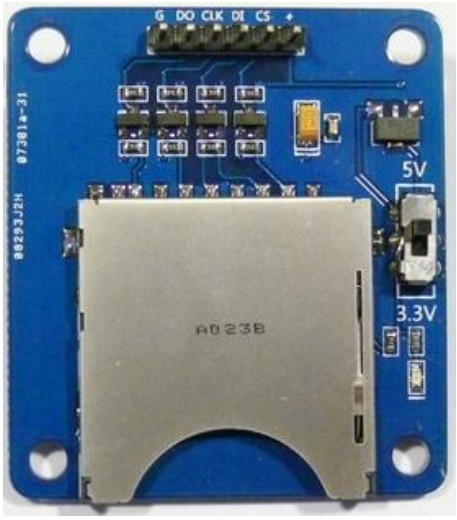
Si on arrête le journal de bord, et que l'on précise que l'on veut utiliser un data logger, un dossier utilisant la date est créé, ainsi qu'un sous dossier utilisant l'heure.

Par exemple .....`\10122015\15H`

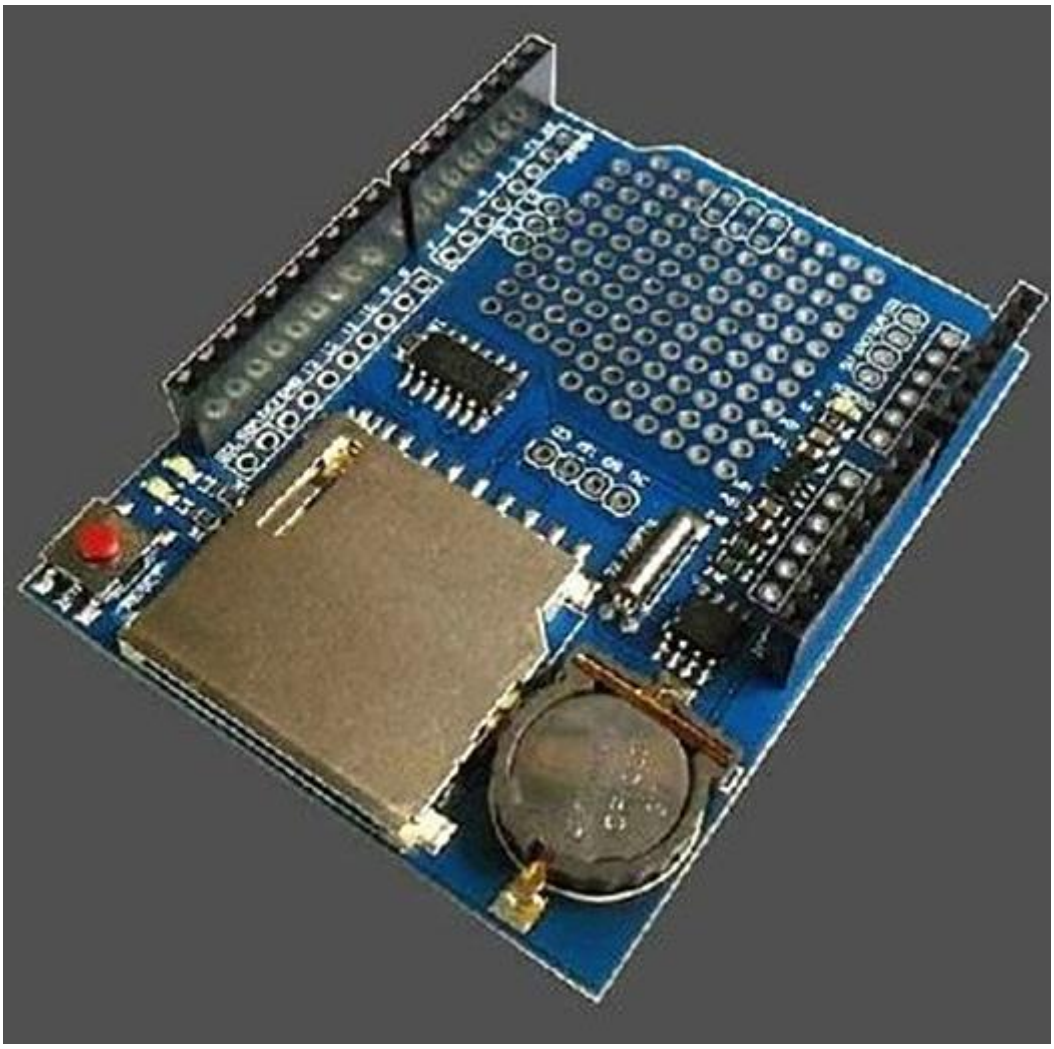
L'équipage, le départ, l'arrivée seront envoyés dans le fichier INIT.txt et les trames NMEA seront envoyées dans le fichier NMEA.txt

Pour utiliser ce programme, il faut au moins une carte Méga

(ne fonctionne pas avec la carte UNO)

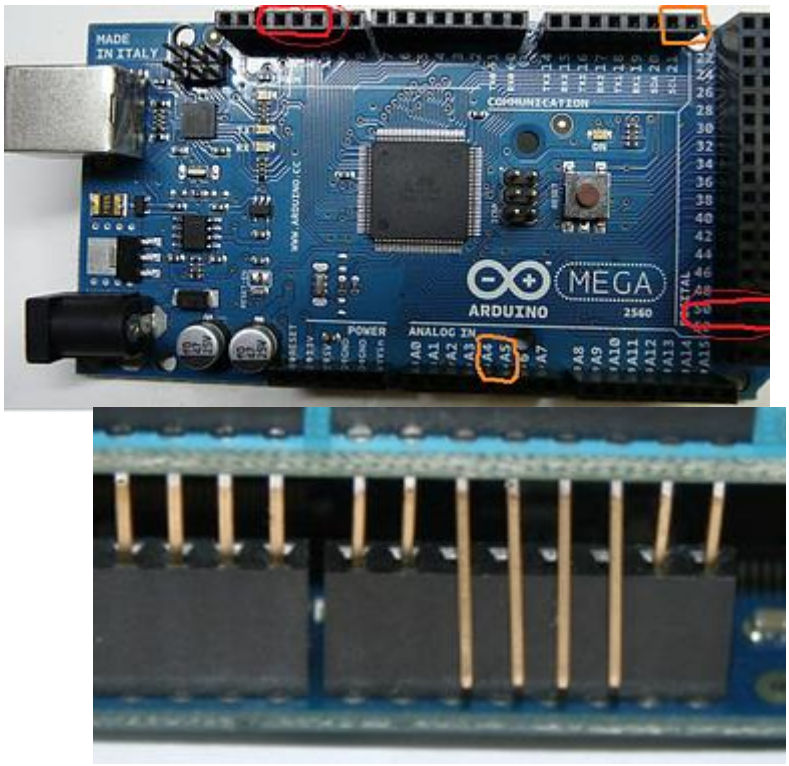


On peut utiliser un module SD et un module RTC



ou un shield SD

Le montage avec le shield : j'ai dû faire des modifications car les shield sont pratiques, mais souvent prévus pour la carte UNO.



La communication entre la carte Arduino et la carte mémoire SD utilise la communication SPI, laquelle utilise les broches 11, 12 et 13 de la plupart des cartes Arduino,

ou les broches 51, 50 et 52 sur la carte Arduino Mega

En plus, une autre broche est utilisée pour sélectionner la carte SD. Cette broche peut être la broche matérielle SS (broche 10 sur la plupart des cartes Arduino et broche 53 sur la Mega)

Il faut donc déporter les broches 10,11,12,13 du shield vers les broches 53,51,50,52 de la carte Méga.

Le module RTC a aussi une connexion différente :

SDA --> pin A4 Data (pin 20 sur Mega)

SCL -->pin A5 Clock (pin 21 sur Mega)

Il faut donc déporter les broches 4 et 5 du shield vers les broches 20 et 21 de la carte Méga.

Le programme :

```
#include <Wire.h>
#include "RTClib.h"
#include <SD.h>

RTC_DS1307 rtc;
const int chipSelect = 53;//10 pour uno
char pathfile[22];   //longueur réelle plus 1
char _phrase[90];
int n;
int _w; // numero du traitement sur le caractere
int _checksum; // somme de controle calculée
int _check;// checksum inclus avec la phrase
int l;
String mastring ;
int charlu;
boolean findeligne;
boolean demlecture;
boolean fini;
char dossier[13];
File myFile;
unsigned long    debutchrono ;
byte demarrer;
byte foiss=0;
DateTime maintenant;

//fonctions
int lecture(char c)
{// si LF et  CR : _w=0 ----->case 0

  if ((c == 10) || (c == 13)) {
    _w = 0;
  }
  if   (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 du caractère '$'
    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
      _w=0;
    }
  default:

    break;
  }
  return 0;
}


char* phrase() {
  // retoune la phrase valide


    return  _phrase;
}


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

}


void ecoutelitenvoie()
{
  int i;
  charlu=0;
  char c;
  mastring="";
  findeligne=false;
```

```
  //debutdeligne=false;
  while ((Serial.available()>0)&&(findeligne==false)   &&(millis() -
debutchrono < 30000 ) )

  {

    charlu=Serial.read();
    delay(1);
    c=char(charlu);

    switch (c)
    {
    case '!':
      fini=true;

      break;


    case '>':
      findeligne=true;

      break;

    default:
      mastring+=c;

    }

  }


  if (findeligne==true)
  {
    int i=0;
    int par1;

    String morceau;
    String reste;


    myFile = SD.open(pathfile, FILE_WRITE);
    reste=mastring;
    while (reste!=""){
      par1=reste.indexOf(')');
      morceau=reste.substring(0,par1);
      reste=reste.substring(par1+1);
      myFile.print(morceau);
      myFile.println();

    }

    myFile .close();
  }

}


void setup()
{
  unsigned int heure ;
  unsigned int minutes;
```

```
  unsigned int secondes;
  int annee;
  unsigned int mois;
  unsigned int jour;
  char datafile[13];
  int test;
  Wire .begin();
  rtc.begin();
  // rtc.adjust(DateTime(__DATE__, __TIME__));
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial2.begin(4800);
  Serial.println("serial1 a 9600, serial2 a 4800");
  Serial.print("Initialisation de la carte SD...");
  pinMode(53, OUTPUT);

  if (!SD.begin(53)) {
    Serial.println("L'initialisation a echoue!");
    return;
  }
  else
  {
    Serial.println("Initialisation reussie.");
    delay(100);
  }

  maintenant = rtc.now();
  annee = maintenant.year();
  jour = maintenant.day();
  mois = maintenant.month();
  heure = maintenant.hour();
  minutes = maintenant.minute();
  //secondes = maintenant.second();
  sprintf(datafile,"%02d%02dA%02dH.TXT",jour,mois,heure);  //  %d pour un
int
  sprintf(dossier,"%02d%02d%04d/%02dH",jour,mois,annee,heure);
  sprintf(pathfile,"%12s/NMEA.TXT",dossier);
  Serial.print("Le dossier du jour :");
  Serial.println(dossier);
  Serial.print("L'heure : ");
  Serial.print(heure);
  Serial.print("h");
  Serial.println(minutes);
  delay(500);
  test = SD.exists(dossier);
  if (test != true)
  {
    Serial.print("Le dossier ");
    Serial.print(dossier);
    Serial.println(" n'existe pas ");
    SD.mkdir(dossier);
    delay(500);
  }
  SD.remove(pathfile);
  myFile = SD.open(pathfile, FILE_WRITE);
  myFile.close();
  n = 0;

  _w = 0;

  _checksum = 0;
```

```
  demlecture=false;
  fini=false;
  mastring="";
  Serial.println("Vous pouvez taper h pour mettre à l'heure");
  Serial.println("Attendre 30 secondes ou rentrer l'initialisation $ pour
commencer ! pour finir ");
  delay(500);
  debutchrono=millis();
  demarrer=0;
  //while (Serial.available()) Serial.read();
}

void loop()
{
  if ((millis() - debutchrono < 30000) && (fini==false))
  {
    //recevoir les fichiers de livre de bord
    while ((demlecture==false)&& (millis() - debutchrono < 30000) )
    {
      if (Serial.available()) charlu=Serial.read();
      if ((demlecture==false)&&(charlu=='h'))
        rtc.adjust(DateTime(__DATE__, __TIME__));
      if (charlu=='$')
      {
        sprintf(pathfile,"%12s/INIT.TXT",dossier);
        SD.remove(pathfile);
        myFile = SD.open(pathfile, FILE_WRITE);
        myFile.close();
        demlecture=true;
      }
    }
    while ((demlecture==true) && (fini==false) && (millis() - debutchrono <
30000))
      ecoutelitenvoie();
  }
  if ((fini==true) || (millis() - debutchrono >= 30000))
  {
    if (demarrer==0)
    {
      Serial.println("(pret)");

      delay(500);
      if (fini==true)
        sprintf(pathfile,"%12s/NMEA.TXT",dossier);
      demarrer=1;
    }



    if (demarrer==1)
    {
      char car = 0; //variable contenant le caractère à lire GPS
      // il y a un caractere à lire ?

      if (Serial1.available())
      {
        maintenant = rtc.now();
        car = Serial1.read(); // lect caractere
        delay(1);
        l=lecture(car);
```

```
        if (l)
        {
          Serial.print(phrase());
          File myFile;
          myFile = SD.open(pathfile, FILE_WRITE);
          myFile.print(maintenant.hour());
          myFile.print(':');
          myFile.print(maintenant.minute());
          myFile.print(':');
          myFile.print(maintenant.second());
          myFile.print(' ');
          myFile.print(phrase());
          myFile .close();
          Serial1.flush();
          if (foiss==0){
            Serial2.end();
            foiss++;
          }
        }
      }
      if (Serial2.available())
      {
        maintenant = rtc.now();
        car = Serial2.read(); // lect caractere
        l=lecture(car);
        if (l)
        {
          Serial.print(phrase());
          File myFile;
          myFile = SD.open(pathfile, FILE_WRITE);
          myFile.print(maintenant.hour());
          myFile.print(':');
          myFile.print(maintenant.minute());
          myFile.print(':');
          myFile.print(maintenant.second());
          myFile.print(' ');
          myFile.print(phrase());
          myFile .close();
          if (foiss==0){
            Serial1.end();
            foiss++;
          }
        }
      }
    }
  }

}
```