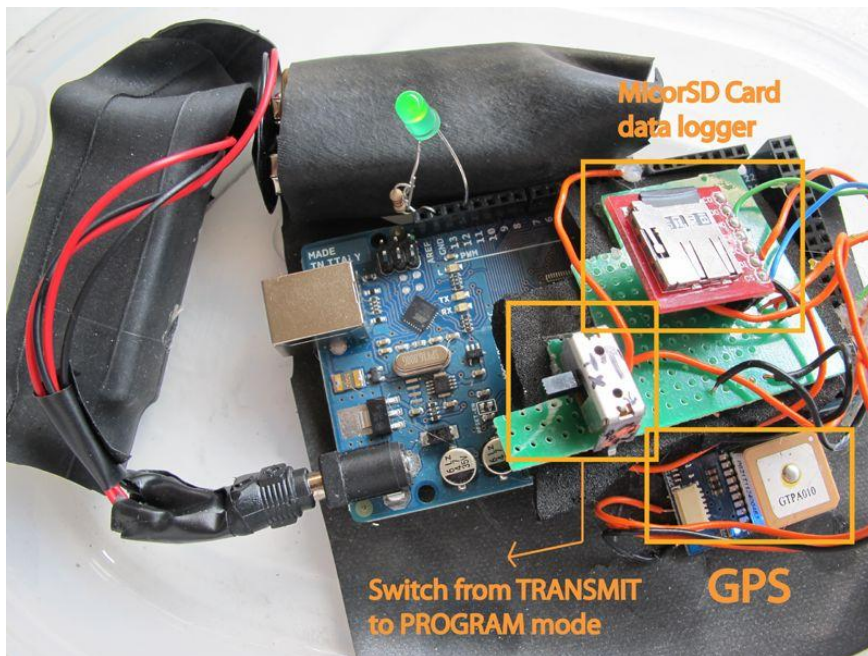


RAPPORT DE PROJET

Traceur de Choc

Jean-Luc KEMAJOU, Lionel HOSSIE



Tuteur de projet : Mr Alexandre BOE, Mr Nicolas DEFRANCE,
Mr. Thomas VANTROYS

Ecole Polytechnique universitaire de Lille (POLYTECH'LILLE)
Département : Informatique – Microélectronique Automatique
4^{ème} année

Sommaire

SOMMAIRE.....	2
PRESENTATION DU PROJET	3
PRESENTATION DU MATERIEL	4
1. Arduino Uno.....	4
2. Arduino ATMEGA 2560.....	5
3. Le module GPS EM406.....	6
4. L'Accéléromètre ADXL345	8
5. Le Shield MicroSD.....	9
TRAVAUX REALISES	10
1. Acquisition des données GPS	10
2. Acquisition des valeurs des accelerations	12
3. Affichage sur l'écran LCD	14
4. Sauvegarde sur la carte SD	15
5. Résultats avec notre traceur obtenu.....	17
6. Réalisation Carte Electronique	18
PROBLEMES RENCONTRES	19
CONCLUSION	20
ANNEXES	21
SOURCES	27

Présentation du projet

Le projet initiation à la recherche est un module obligatoire pour pouvoir valider notre Semestre 8 à Polytech'Lille. Ce projet doit permettre à nous élèves-ingénieurs de nous initier à la recherche avec un long travail de documentation et de réflexion.

Notre choix s'est porté sur le traceur de choc, le but étant de réaliser une carte permettant de mesurer l'accélération, intégrant un GPS. D'une manière plus explicite, cette carte devrait nous permettre dans un premier temps de tracer par exemple la position d'un colis. Grâce au GPS nous devrions être capables de donner la position du colis à tout moment (Géo localisation) mais par contre en cas de perte du signal, nous devrions passer par l'accéléromètre pour estimer à nouveau la position du colis. Dans un second temps, nous devons enregistrer dans une carte SD les différentes valeurs des accélérations ainsi que la position (latitude, longitude, date, heure) afin de savoir ou d'estimer si il y a eu un choc ou pas et la position ou cela a eu lieu. Enfin, nous devons être capable si le temps le permet, de transmettre en temps réel ces données à la fin à une montre TI via Liaison radio. L'ensemble du matériel nécessaire à ce projet a été fourni par l'école et une partie a été achetée.

Dans ce rapport, nous commencerons par vous présenter le matériel mis à notre disposition pour pouvoir réaliser notre projet. Dans un deuxième temps, le travail réalisé tout au long de ces 10 semaines sera abordé en détail. Nous continuerons avec les différents problèmes rencontrés et les solutions que nous avons apportés pour en résoudre une partie. On conclura en tirant un bilan de ce projet, et on évoquera l'influence de ce dernier sur le choix probable de l'orientation que nous envisageons de donner à notre carrière professionnelle.

Matériel Requis :

- ✓ Un Arduino UNO
- ✓ Un Arduino ATMEGA 2560
- ✓ Un Module GPS (NMEA GPS)
- ✓ Un Accéléromètre(ADXL345)
- ✓ Un afficheur LCD (Shield LCD couleur)
- ✓ Une Carte SD

Matériel Acheté :

- ✓ Pile pression 9V
- ✓ Connecteur pression

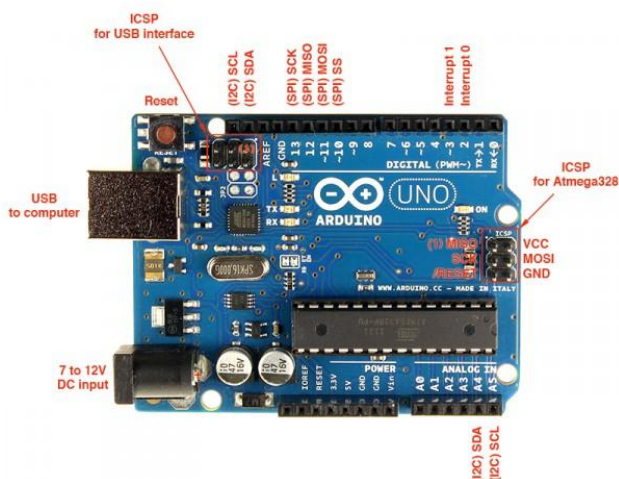
Présentation du matériel**1. Arduino Uno**

C'est la carte arduino « de base », la plus classique qui soit. Elle appartient à la famille des cartes arduino dites « classiques » elle est basée généralement sur le même microcontrôleur AVR à savoir un ATmega328p du fabricant ATMEL.

Par conséquent toutes les cartes utilisant ce microcontrôleur ont les mêmes caractéristiques avec, selon la carte, quelques bonus en plus.

Voici les spécifications pour toutes les cartes arduino classiques à base d'ATmega328p :

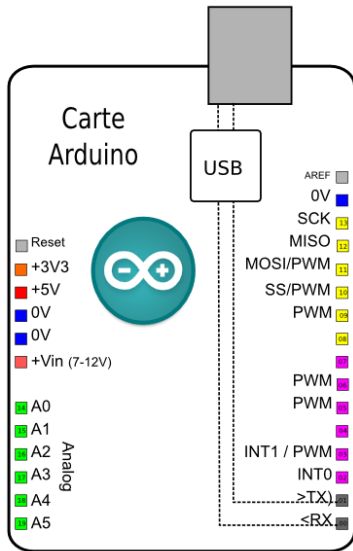
En général :



- ✓ - CPU 8 bits à 16MHz (ou 8MHz pour les cartes 3v3)
- ✓ - Niveau logique 0v / 5v (ou 0v / 3v3 pour les cartes 3v3),
- ✓ - 32Ko de mémoire pour le programme, mais « seulement » 30Ko réellement utilisables,
- ✓ - 2Ko de mémoire
- ✓ - 1Ko de EEPROM (mémoire de stockage non volatile).

Au niveau des entrées / sorties :

- 14 entrées / sorties numériques (« 0" ou « 1" logique) dont 6 sorties PWM



- 6 entrées analogiques (pouvant cependant être utilisé en entrées / sorties logiques si besoin)
- 40mA maximum sur une même broche
- 200mA au total sur toute les broches numérique et analogique

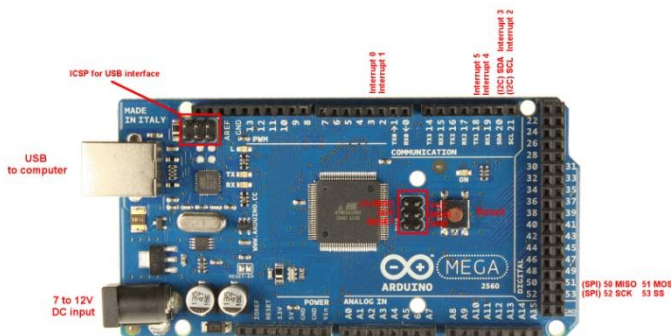
Certaines broches ont des fonctionnalités spéciales :

- D0 et D1 : port série TTL,
- D10, D11, D12, et D13 : bus / port SPI,
- A4 et A5 : bus I2C.

2. Arduino ATMEGA 2560

C'est une carte arduino officielle mais avec beaucoup plus d'entrées / sorties et 2 fois plus de mémoire qu'une carte arduino UNO. L'Arduino MEGA est la plus puissante de la famille Arduino, elle est équipée du microcontrôleur ATmega2560.

Les plus :



- ✓ - 54 entrées / sorties numériques dont 14 avec PWM,
- ✓ - 16 entrées analogiques,
- ✓ - 256Ko de mémoire flash (248Ko réellement utilisable), 8Ko de SRAM et 4Ko d'EEPROM,
- ✓ - 4 port séries hardware,
- ✓ - 6 interruptions extérieures (au lieu de 2).

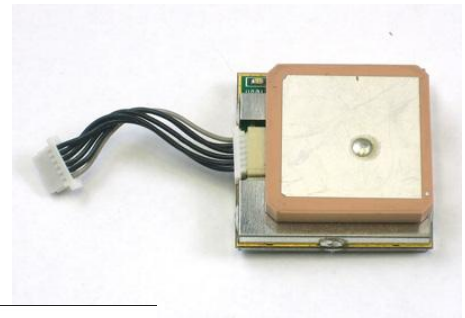
Les moins :

- ✓ - brochage spéciale incompatible avec certains shields arduino,
- ✓ - spécificité demandant une certaine connaissance en programmation,
- ✓ - pas fait pour les débutants,
- ✓ - port I2C et SPI sur des broches différentes d'une carte arduino classique.

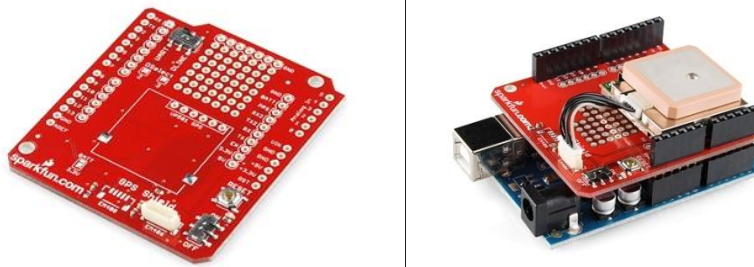
3. Le module GPS EM406

Le module GPS EM-406 de « USGlobalSat » est basé la spectaculaire puce « SiRF StarIII ». Ce module complet est construit sur la même technologie que l'ET-301, mais comprend réglementation tension de bord, indicateur LED d'état, RAM sauvegardée par pile, et une antenne patch! Câble d'interface 6 broches inclus.

Il s'utilise de façon couplée avec le shield gps de Sparkfun, qui permet et l'utilisation simplifiée de ce module GPS avec une carte Arduino.

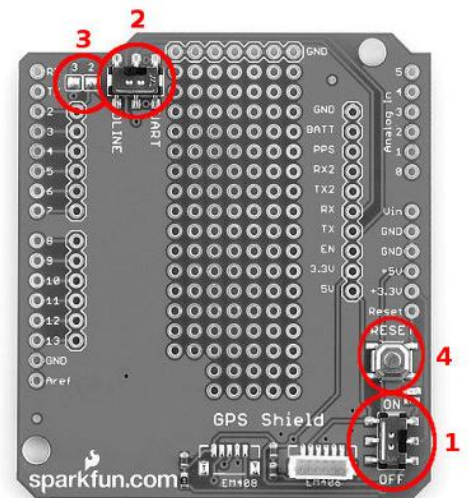


sur
la



Sur le schéma ci-contre, nous pouvons bien observer les différents éléments du shield GPS :

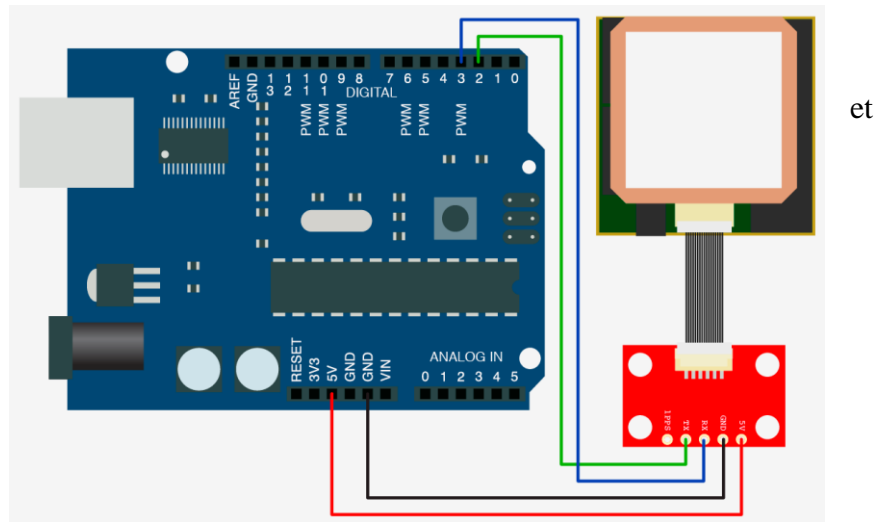
- ✓ 1 : est le bouton d'alimentation qui permet d'alimenter notre GPS.
- ✓ 2 : est l'interrupteur de sélection de l'UART: Si l'UART est sélectionné, le GPS communiquera avec l'arduino directement avec les broches 0 et 1. si DTLINE est sélectionné, le GPS sera connecté par défaut aux broches 2 et 3. DLINE doit donc être d'abord sélectionné pour téléverser le programme dans l'arduino car le mode UART utilise les mêmes broches utilisées pour programmer l'arduino. si le mode UART est sélectionné et que par la suite nous téléversons le code, nous aurons des erreurs dans l'IDE de l'arduino qui signalera un Conflit de bus.
- ✓ 3: Cavaliers à Souder
- ✓ 4: Bouton reset: ce bouton est directement relié au bouton reset de l'arduino donc on peut appuyer sur ce bouton pour relancer notre programme.



❖ Brochage

Ce shield assure simplement la connexion entre le connecteur du module GPS et les broches 0(RX)/1(TX) ou 2(TX soft) et 3 (RX soft) de la carte Arduino.

Un simple inverseur permet le choix entre les 2 modes de communication matériel (broches 0 1) ou logiciel (broches 2 et 3).



❖ Fonctionnement :

L'antenne du module reçoit le signal en provenance des satellites GPS, le décode et le transforme en une chaîne de caractère qui est émise sous la forme d'une communication série.

La chaîne de caractère renvoyée par le module (ou trame GPS) est au format NMEA, ce qui donne quelque chose de la forme :

\$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,0000*0E .

Le module envoie à intervalle régulier des chaînes de caractères, appelées trames GPS qui contiennent l'information issue des satellites (longitude, latitude, date, jour, heure, etc...)

Par défaut, le module envoie en permanence tous les types de trames.

Heureusement, il est possible d'envoyer des instructions au module pour fixer la fréquence d'envoi des trames, le type de trame, stopper l'envoi des trames, d'obtenir une trame à la demande, etc...

A noter que la librairie « TinyGPS » fournit les fonctions pour le décodage des trames GPS.

Les types de trames disponibles sont :

- ✓ La trame GGA (Global Positioning System Fixed Data)
- ✓ La trame GLL (Geographic Position-Latitude/Longitude) : Trame simplifiée pour la latitude et la longitude.
- ✓ La trame GSA : Cette trame nous donne les infos sur les satellites utilisés
- ✓ La trame GSV : Cette trame nous donne les infos sur les satellites en vue
- ✓ La trame RMC (Recommended Minimum Specific GNSS Data) : trame recommandée
- ✓ La trame VTG : Calcul de vitesse

Les principales trames que nous utiliserons pour notre projet seront la trame **GGA** et la trame **GLL** des exemples d'utilisation sont présentée ci-contre :

GGA-Global Positioning System Fixed Data

Table B-2 contains the values for the following example:

\$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1,0,9,0,M,,,,,0000*18

GLL-Geographic Position-Latitude/Longitude

Table B-4 contains the values for the following example:

\$GPGLL,3723.2475,N,12158.3416,W,161229.487,A*2C

Table B-2 GGA Data Format

Name	Example	Units	Description
Message ID	\$GPGGA		GGA protocol header
UTC Time	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Position Fix Indicator	1		See Table B-3
Satellites Used	07		Range 0 to 12
HDOP	1.0		Horizontal Dilution of Precision
MSL Altitude ¹	9.0	meters	
Units	M	meters	
Geoid Separation ¹		meters	
Units	M	meters	
Age of Diff. Corr.		second	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*18		
<CR><LF>			End of message termination

Table B-4 GLL Data Format

Name	Example	Units	Description
Message ID	\$GPGLL		GLL protocol header
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	n		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
UTC Position	161229.487		hhmmss.sss
Status	A		A=data valid or V=data not valid
Checksum	*2C		
<CR><LF>			End of message termination

Le Format d'instruction de contrôle du module GPS est le suivant :

\$PSRF103, <msg>, <mode>, <rate>, <cksumEnable>*CKSUM<CR><LF> avec :

- ✓ <msg> : 00=GGA,01=GLL,02=GSA,03=GSV,04=RMC,05=VTG // le type de trame concerné
- ✓ <mode> : 0=SetRate,1=Query // requete ou config débit
- ✓ <rate> : Output every <rate>seconds, off=0,max=255 // valeur débit - 0 pour off
- ✓ <cksumEnable> : 00=disable Checksum,01=Enable checksum for specified message // code de contrôle.
- ✓ <CR><LF> : Hex 0D 0A

4. L'Accéléromètre ADXL345

L'ADXL345 est un accéléromètre petit, mince, de faible puissance, à 3 axes avec une résolution élevée (13-bit) qui mesure jusqu'à ± 16 g. Les données de sortie numérique sont accessibles soit par un **SPI** ou interface numérique **I2C**.

L'ADXL345 est bien adapté pour mesurer l'accélération statique de la gravité dans les applications d'inclinaison-détection, ainsi que l'accélération dynamique résultant du mouvement de choc. Sa haute résolution (4 mg / LSB) permet de mesurer les changements d'inclinaison inférieure à 1,0 °.

L'excellente sensibilité (3.9mg/LSB @ 2g) offre une sortie de précision jusqu'à ± 16g.

Comme nous l'avons dit plus haut, il existe deux protocoles de communication pour l'ADXL345, le protocole SPI et le I2C.



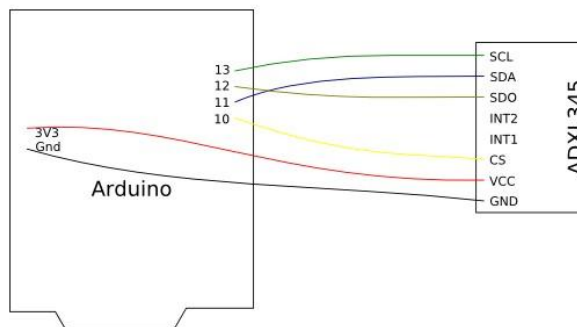
haute

protocole

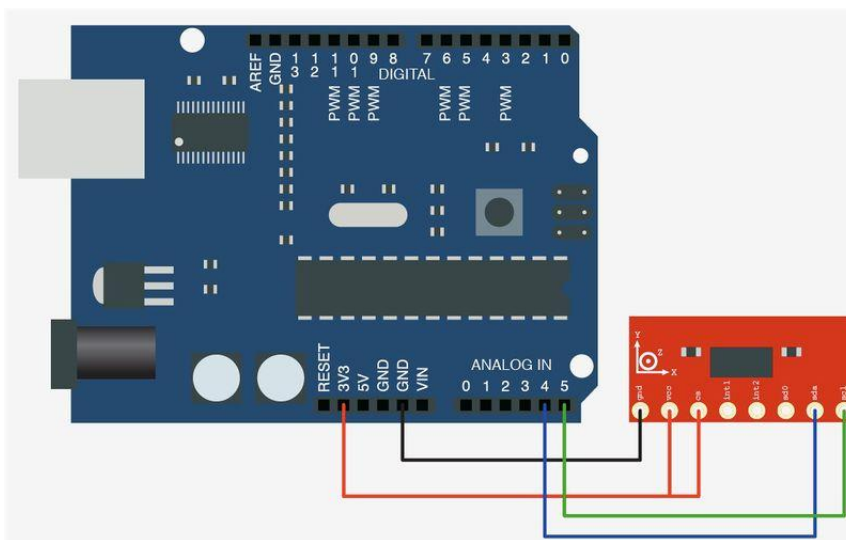
Arduino Pin	ADXL345 Pin
10	CS
11	SDA
12	SDO
13	SCL
3V3	VCC
Gnd	GND

Brochage Mode SPI :

Here is a diagram in case you like pictures!



Brochage Mode I2C :



5. Le Shield MicroSD

Ce petit module est destiné à venir s'enficher sur la platine Arduino ou Arduino MEGA afin de vous permettre de disposer d'une solution de stockage sur carte microSD. La communication entre le module et la platine Arduino s'effectue via un bus SPI. Les broches SCK, DI et DO de la prise microSD sont réparties sur l'ATmega168/328 standard sur les broches SPI (numérique 11-13), tandis que la broche CS est connectée au l'arduino.

Avec ce module microSD pour Arduino, on très grande quantité d'espace de stockage des module nous permet de brancher une carte microSD Standard memory) et d'écrire / lire des données dans celui-ci.



ajoute une données. Ce (carte Flash

Travaux Réalisés

Nous tenons tout d'abord à mentionner que nous avons fait tout notre programme dans un premier temps sur un « Arduino UNO » puis par la suite, à cause des problèmes que nous avons rencontrés au cours du projet et que nous vous expliquerons plus tard, nous avons décidé d'implanter notre programme plutôt dans un « Arduino ATMEGA2560 ». Dans la suite de cette partie, nous ne parlerons que des travaux réalisés sur arduino « ATMEGA2560 ».

1. Acquisition des données GPS

Dans un premier temps nous avons récupéré les données fournies par notre GPS, à savoir, la latitude, la longitude, l'heure, la date, etc..

Voici le code qui nous a permis de faire cette acquisition :

```
*****  
  
#include <TinyGPS.h>  
  
/* This sample code demonstrates the normal use of a TinyGPS object.  
   It requires the use of SoftwareSerial, and assumes that you have a  
   4800-baud serial GPS device hooked up on pins RX3 and TX3 for (serial3).  
*/  
  
TinyGPS gps;  
  
static void gpstdump(TinyGPS &gps);  
static bool feedgps();  
static void print_float(float val, float invalid, int len, int prec);  
static void print_int(unsigned long val, unsigned long invalid, int len);  
static void print_date(TinyGPS &gps);  
static void print_str(const char *str, int len);  
  
*****  
  
void setup()  
{  
  Serial.begin(115200);  
  Serial3.begin(4800);  
}  
*****
```

```

*****
void loop()
{
  bool newdata = false;
  unsigned long start = millis();

  // Every second we print an update
  while (millis() - start < 1000)
  {
    if (feedgps())
      newdata = true;
  }

  gpsdump(gps);
}
*****

static void gpsdump(TinyGPS &gps)
{
  float flat, flon;
  unsigned long age, date, time, chars = 0;
  unsigned short sentences = 0, failed = 0;
  static const float PARIS_LAT = 48.85661, PARIS_LON = 2.35222;
  print_int(gps.satellites(), TinyGPS::GPS_INVALID_SATELLITES, 5);
  print_int(gps.hdop(), TinyGPS::GPS_INVALID_HDOP, 5);
  gps.f_get_position(&flat, &flon, &age);
  print_float(flat, TinyGPS::GPS_INVALID_F_ANGLE, 9, 5);
  print_float(flon, TinyGPS::GPS_INVALID_F_ANGLE, 10, 5);
  print_int(age, TinyGPS::GPS_INVALID_AGE, 5);
  dtostrf(flon, 7, 6, lcd_long);
  dtostrf(flat, 7, 6, lcd_lat);
  //sprintf(lcd_long, "%d", flon*100000);
  // sprintf(lcd_lat, "%d", flat*100000);
  print_date(gps);
  print_float(gps.f_altitude(), TinyGPS::GPS_INVALID_F_ALTITUDE, 8, 2);
  print_float(gps.f_course(), TinyGPS::GPS_INVALID_F_ANGLE, 7, 2);
  print_float(gps.f_speed_kmph(), TinyGPS::GPS_INVALID_F_SPEED, 6, 2);
  print_str(gps.f_course() == TinyGPS::GPS_INVALID_F_ANGLE ? "**** " :
TinyGPS::cardinal(gps.f_course()), 6);
  print_int(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0UL : (unsigned
long)TinyGPS::distance_between(flat, flon, PARIS_LAT, PARIS_LON) / 1000, 0xFFFFFFFF, 9);
  print_float(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : TinyGPS::course_to(flat, flon,
51.508131, -0.128002), TinyGPS::GPS_INVALID_F_ANGLE, 7, 2);
  print_str(flat == TinyGPS::GPS_INVALID_F_ANGLE ? "**** " :
TinyGPS::cardinal(TinyGPS::course_to(flat, flon, PARIS_LAT, PARIS_LON)), 6);
  gps.stats(&chars, &sentences, &failed);
  print_int(chars, 0xFFFFFFFF, 6);
  print_int(sentences, 0xFFFFFFFF, 10);
  print_int(failed, 0xFFFFFFFF, 9);
  // Serial.println();
}
*****
*****

```

```

static bool feedgps()
{
  while (Serial3.available())
  {
    if (gps.encode(Serial3.read()))
      return true;
  }
  return false;
}
*****

```

Explications du code: toutes les secondes on récupère les coordonnées GPS par le port série. La fonction « gpsdump » reformate les données issues du gps pour l’afficher de façon plus présentable sur le moniteur série.

Résultats Obtenus: voir figure1 de l’annexe

2. Acquisition des valeurs des accelerations

Après avoir pu recueillir nos informations provenant du GPS, il était question de récupérer les valeurs d’accélération. Donc pour ce faire, nous avons branché l’accéléromètre en communication I2C, ensuite nous avons téléverser le coder suivant :

```

*****
#include <Wire.h>

#define DEVICE (0x53) //ADXL345 device address
#define TO_READ (6) //num of bytes we are going to read each time (two bytes for each axis)

byte buff[TO_READ] ; //6 bytes buffer for saving data read from the device
char acc[512];

*****
*****

void setup()
{
  Serial.begin(115200);
  Wire.begin(); // join i2c bus (address optional for master)
  pinMode(10, OUTPUT);

  //Turning on the ADXL345
  writeTo(DEVICE, 0x2D, 0);
  writeTo(DEVICE, 0x2D, 16);
  writeTo(DEVICE, 0x2D, 8);
}

```

```

*****
void loop()
{
    readFrom(DEVICE, regAddress, TO_READ, buff); //read the acceleration data
from the ADXL345
    //each axis reading comes in 10 bit resolution, ie 2 bytes. Least Significat Byte first!!
    //thus we are converting both bytes in to one int
    x = (((int)buff[1]) << 8) | buff[0];
    y = (((int)buff[3])<< 8) | buff[2];
    z = (((int)buff[5]) << 8) | buff[4];

    //we send the x y z values as a string to the serial port
    sprintf(acc, "%02d\t%02d\t%02d", x, y, z);
    Serial.println(acc);
}
*****
void writeTo(int device, byte address, byte val) {
    Wire.beginTransmission(device); //start transmission to device
    Wire.write(address); // send register address
    Wire.write(val); // send value to write
    Wire.endTransmission(); //end transmission
}
void readFrom(int device, byte address, int num, byte buff[]) {
    Wire.beginTransmission(device); //start transmission to device
    Wire.write(address); //sends address to read from
    Wire.endTransmission(); //end transmission
    Wire.beginTransmission(device); //start transmission to device (initiate again)
    Wire.requestFrom(device, num); // request 6 bytes from device
    int i = 0;
    while(Wire.available()) //device may send less than requested (abnormal)
    {
        buff[i] = Wire.read(); // receive a byte
        i++;
    }
    Wire.endTransmission(); //end transmission
}
*****

```

Explication du code : Les fonctions « writeTo » et « ReadFrom » assure la comminication en mode I2C. Les valeurs de l'accélération en x, y, z sont stockés dans un buffer (de taille 6 bytes).

Puis ce buffer est décomposé en 3 entiers x y z :

```

x = (((int)buff[1]) << 8) | buff[0];
y = (((int)buff[3])<< 8) | buff[2];
z = (((int)buff[5]) << 8) | buff[4];

```

Correspondant à l'accélération sur chaque axe.

Résultats Obtenus: voir figure1 de l'annexe

3. Affichage sur l'écran LCD

Maintenant, ayant déjà pu avoir les données du GPS et ceux de l'accéléromètre, il était question de les afficher sur l'écran LCD présenté un plus haut. Le bout de code suivant vous montre le principe d'affichage sur l'écran, ainsi que la bibliothèque et les commandes utilisées pour le faire.

```
*****
#include <gLCD.h>
const char RST = 8;
const char CS = 9;
const char Clk = 13;
const char Data = 11;
gLCD graphic(RST,CS,Clk,Data,HIGH_SPEED);
char lcd_date[512];
char lcd_time[512];
char lcd_long[512];
char lcd_lat[512];
int cpt_choc;
#define BACKGROUND BLACK // room for growth, adjust the background color according to daylight
#define FONT WHITE

*****
void setup()
{ graphic.begin(0,0,0,PHILLIPS_0);
  graphic.setBackColour(15,15,15);
  graphic.setForeground(0,0,0);
  graphic.setFont(Normal_SolidBG);
  graphic.setCoordinate(5,0);
  graphic.print("TRACEUR DE CHOC");
  graphic.setCoordinate(5,10);
  graphic.print(" BY HOSSIE & KEMAJOU");
}

*****
*****
Void loop()
{
graphic.setCoordinate(70,30);
  graphic.print(lcd_date);

  graphic.setCoordinate(70,40);
  graphic.print(lcd_time);

  graphic.setCoordinate(70,50);
  graphic.print(lcd_long);

  graphic.setCoordinate(70,60);
  graphic.print(lcd_lat);

  graphic.setCoordinate(60,70);
  graphic.print(acc);
}
```

```

graphic.setCoordinate(60,80);
  graphic.print(cpt_choc);
}

```

Explications du code: la librairie gLCD.h nous donne des primitives très faciles à utiliser qui nous permettent d'afficher de chaînes de caractères sur notre shield LCD. Nous affichons à chaque boucle du programme les coordonnées gps , les accélérations , un compteur de choc et « choc »(si un choc a été détecté).

4. Sauvegarde sur la carte SD

Le but principal de notre application étant de retracer les chocs qu'un colis subirait lors de son transport, nous décidons d'enregistrer les coordonnées GPS, l'heure, la date et les accélérations à chaque fois qu'un choc est détecté.

Nous avons décidé de considérer que notre colis subit un choc si l'une des accélérations est supérieur à 2,4g (soit 300 après conversion analogique-numérique).

```

#include <SPI.h>
#include <SD.h>
File myFile;
void setup()
{ pinMode(10, OUTPUT);
  if (!SD.begin(53)) {
    graphic.print("initialization failed!");
  }
  graphic.print("initialization done.");
  myFile = SD.open("logger.txt", FILE_WRITE);

  if (myFile) {
    myFile.println("TRACEUR DE CHOC");
    // close the file:
    myFile.close();
  } else {
    // if the file didn't open, print an error:
    graphic.println(" error opening file");
  }
}
void loop()
{
  if ((abs(x)>300)||abs(y)>300)||abs(z)>300)
  {
    cpt_choc++;
    graphic.setCoordinate(40,100);
    graphic.print("#####CHOC#####");
    delay(1000);
    graphic.setForeground(15,15,15);
  }
}

```

```
graphic.print(" ");
graphic.setForeground(0,0,0);
myFile = SD.open("logger.txt", FILE_WRITE);
if (myFile) {
  Serial.println("#####CHOC#####");
  myFile.println("-----");
  myFile.println("#####CHOC#####");
  myFile.print("DATE: ");
  myFile.println(lcd_date);
  myFile.print("TIME: ");
  myFile.println(lcd_time);
  myFile.print("LONGITUDE: ");
  myFile.println(lcd_long);
  myFile.print("LATITUDDE: ");
  myFile.println(lcd_lat);
  myFile.println("-----");
  myFile.close();
} else {
  graphic.setCoordinate(5,80);
  graphic.println("error opening test.txt");
  // if the file didn't open, print an error:
  Serial.println("error opening test.txt");
}
*****
```


5. Résultats avec notre traceur obtenu.

Voici donc un imprime-écran des résultats obtenus sur le moniteur série et dans la micro SD.

Traceur de choc 12
by HOSSIE & KEMAJOU IMA4 SC 2012/2013
Sizeof(ppsobject) = 115

Sats	HDOF	Latitude (deg)	Longitude (deg)	Fix Age	Date	Time	Date Alt Age (m)	Course	Speed	Card	Distance	Course Card	Chars RX	Sentences RX	Checksum Fail	acceleration x	acceleration y	acceleration z	
5	300	50.60777	3.13701	244	05/07/2013	09:07:24	274 50.50	266.38	0.11	W	202	294.95 SSW	260	2	0	30	50	229	
5	300	50.60776	3.13702	371	05/07/2013	09:07:25	400 51.10	151.82	1.61	SSE	202	294.95 SSW	457	4	0	32	61	227	
5	300	50.60776	3.13703	786	05/07/2013	09:07:26	811 51.60	151.82	1.61	SSE	202	294.95 SSW	832	5	0	18	61	233	
5	300	50.60776	3.13703	627	05/07/2013	09:07:27	656 51.80	329.82	2.07	HNW	202	294.95 SSW	1046	8	0	23	55	217	
5	300	50.60776	3.13703	749	05/07/2013	09:07:28	779 52.00	320.17	3.43	HNW	202	294.95 SSW	1294	10	0	32	58	226	
5	300	50.60776	3.13703	169	05/07/2013	09:07:30	195 51.50	317.59	3.50	HNW	202	294.95 SSW	1546	13	0	37	88	211	
5	300	50.60777	3.13702	297	05/07/2013	09:07:31	321 51.00	320.72	5.59	HNW	202	294.95 SSW	1798	15	0	27	71	230	
5	300	50.60778	3.13700	137	05/07/2013	09:07:32	166 50.80	317.83	7.98	HNW	202	294.95 SSW	2227	18	0	21	84	223	
5	300	50.60779	3.13699	257	05/07/2013	09:07:33	287 51.00	312.74	5.57	HNW	202	294.95 SSW	2424	20	0	35	70	206	
5	300	50.60779	3.13700	391	05/07/2013	09:07:34	420 50.70	303.74	2.83	HNW	202	294.95 SSW	2621	22	0	36	43	233	
5	300	50.60776	3.13700	513	05/07/2013	09:07:35	543 50.60	288.27	1.20	W	202	294.95 SSW	2815	24	0	33	33	245	
5	300	50.60776	3.13701	191	05/07/2013	09:07:36	221 50.70	212.43	0.69	SSW	202	294.95 SSW	3211	26	0	66	-13	227	
5	300	50.60776	3.13702	766	05/07/2013	09:07:37	796 50.60	165.58	1.69	SSE	202	294.95 SSW	3465	28	0	51	37	221	
6	240	50.60774	3.13707	188	05/07/2013	09:07:39	213 51.10	148.84	1.59	SSE	202	294.95 SSW	3721	31	0	48	05	229	
6	240	50.60779	3.13710	311	05/07/2013	09:07:40	337 51.20	136.64	0.13	SE	202	294.95 SSW	3974	33	0	31	19	243	
6	240	50.60772	3.13711	439	05/07/2013	09:07:41	465 51.20	137.11	1.56	SE	202	294.95 SSW	4229	35	0	28	24	238	
6	240	50.60771	3.13711	276	05/07/2013	09:07:42	306 51.10	97.87	1.44	E	202	294.95 SSW	4602	38	0	17	32	243	
6	240	50.60770	3.13712	394	05/07/2013	09:07:43	423 51.10	87.05	1.81	E	202	294.95 SSW	4800	40	0	26	21	220	
6	240	50.60769	3.13715	529	05/07/2013	09:07:44	559 50.60	87.13	2.44	E	202	294.95 SSW	4998	42	0	34	19	224	
6	240	50.60768	3.13717	654	05/07/2013	09:07:45	684 49.80	103.00	1.80	ESE	202	294.95 SSW	5206	44	0	35	32	226	
6	240	50.60767	3.13719	321	05/07/2013	09:07:46	350 49.50	86.38	1.91	E	202	294.95 SSW	5660	46	0	34	39	228	
5	290	50.60766	3.13721	206	05/07/2013	09:07:48	231 48.60	97.65	2.32	E	202	294.95 SSW	5915	49	0	33	-1	218	
5	290	50.60766	3.13722	337	05/07/2013	09:07:49	363 46.90	114.64	1.19	ESE	202	294.95 SSW	6169	51	0	35	16	226	
5	290	50.60765	3.13723	168	05/07/2013	09:07:50	197 45.10	136.03	1.87	SE	202	294.95 SSW	6384	54	0	34	23	246	
5	290	50.60765	3.13723	590	05/07/2013	09:07:51	615 43.70	136.03	1.87	SE	202	294.95 SSW	6673	55	0	26	20	235	
4	740	50.60765	3.13723	431	05/07/2013	09:07:52	461 43.50	272.79	1.50	W	202	294.95 SSW	6975	58	0	15	59	232	
4	740	50.60765	3.13723	562	05/07/2013	09:07:53	592 43.10	308.46	1.06	NW	202	294.95 SSW	7170	60	0	30	36	229	
4	740	50.60765	3.13722	683	05/07/2013	09:07:54	713 43.30	284.91	2.56	HNW	202	294.95 SSW	7386	62	0	14	50	237	
4	740	50.60766	3.13720	817	05/07/2013	09:07:55	847 43.40	299.82	2.48	HNW	202	294.95 SSW	7636	65	0	21	63	226	
4	740	50.60767	3.13717	231	05/07/2013	09:07:57	256 43.40	299.82	2.48	HNW	202	294.95 SSW	8000	66	0	-63	158	404	
*****CHOC*****																			
4	740	50.60767	3.13715	171	05/07/2013	09:07:59	201 43.40	322.98	2.39	HNW	202	294.95 SSW	8261	68	1	10	65	210	
5	290	50.60767	3.13715	297	05/07/2013	09:08:00	326 43.90	319.96	1.65	NW	202	294.95 SSW	8458	70	1	06	71	218	
5	290	50.60767	3.13715	714	05/07/2013	09:08:01	739 44.50	319.96	1.65	NW	202	294.95 SSW	8802	71	1	01	76	216	
5	290	50.60767	3.13714	546	05/07/2013	09:08:02	575 45.10	311.32	2.26	NW	202	294.95 SSW	9052	74	1	-51	-47	282	
5	290	50.60766	3.13715	684	05/07/2013	09:08:03	714 45.80	268.80	1.19	W	202	294.95 SSW	9270	76	1	-38	-52	-193	
5	290	50.60766	3.13717	108	05/07/2013	09:08:05	133 46.40	309.85	0.63	NW	202	294.95 SSW	9525	79	1	23	56	220	

logger - Bloc-notes

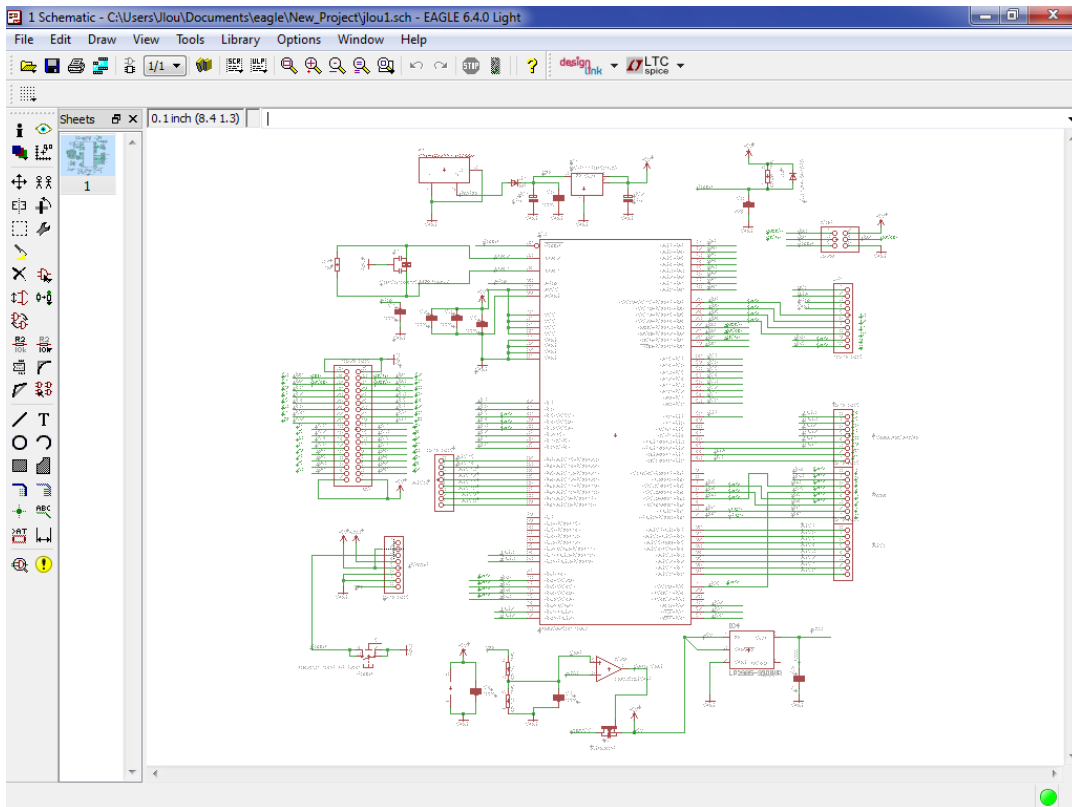
Traceur de choc

```

*****CHOC*****
DATE:
TIME:
LONGITUDE: 1000.000000
LATITUDE: 1000.000000
ACCELERATION: 104 321 566
-----
*****CHOC*****
DATE: 05/07/2013
TIME: 08:53:40
LONGITUDE: 3.137200
LATITUDE: 50.607201
ACCELERATION: -63 158 404
-----
Traceur de choc
*****CHOC*****
DATE: 05/07/2013
TIME: 08:54:51
LONGITUDE: 3.137260
LATITUDE: 50.607368
ACCELERATION: -144 -711 441
-----
Traceur de choc
*****CHOC*****
DATE: 05/07/2013
TIME: 08:55:28
LONGITUDE: 3.137210
LATITUDE: 50.607681
ACCELERATION: 145 55 781
-----
Traceur de choc
*****CHOC*****
DATE: 05/07/2013
TIME: 08:57:05
LONGITUDE: 3.137100
LATITUDE: 50.607712
ACCELERATION: -475 25 97
-----
*****CHOC*****
DATE: 05/07/2013
TIME: 08:57:26
LONGITUDE: 3.137120
LATITUDE: 50.607712
ACCELERATION: -54 547 94
-----
*****CHOC*****
DATE: 05/07/2013
TIME: 08:57:48
LONGITUDE: 3.137140
LATITUDE: 50.607700
ACCELERATION: -458 -01 475
-----
Traceur de choc
    
```

6. Réalisation Carte Electronique

Ici l'objectif était de concevoir une version miniaturisée de notre traceur autour d'un micro processeur Atmel. En effet dans la première version, le module arduino certes stable et performant est assez volumineux car comportait des éléments dont nous n'avons pas besoin pour notre projet (port usb, broches analogiques). Nous avons donc décidé de partir sur les base du schématique d'un arduino UNO, en ne conservant que les éléments qui nous étaient vraiment utile.



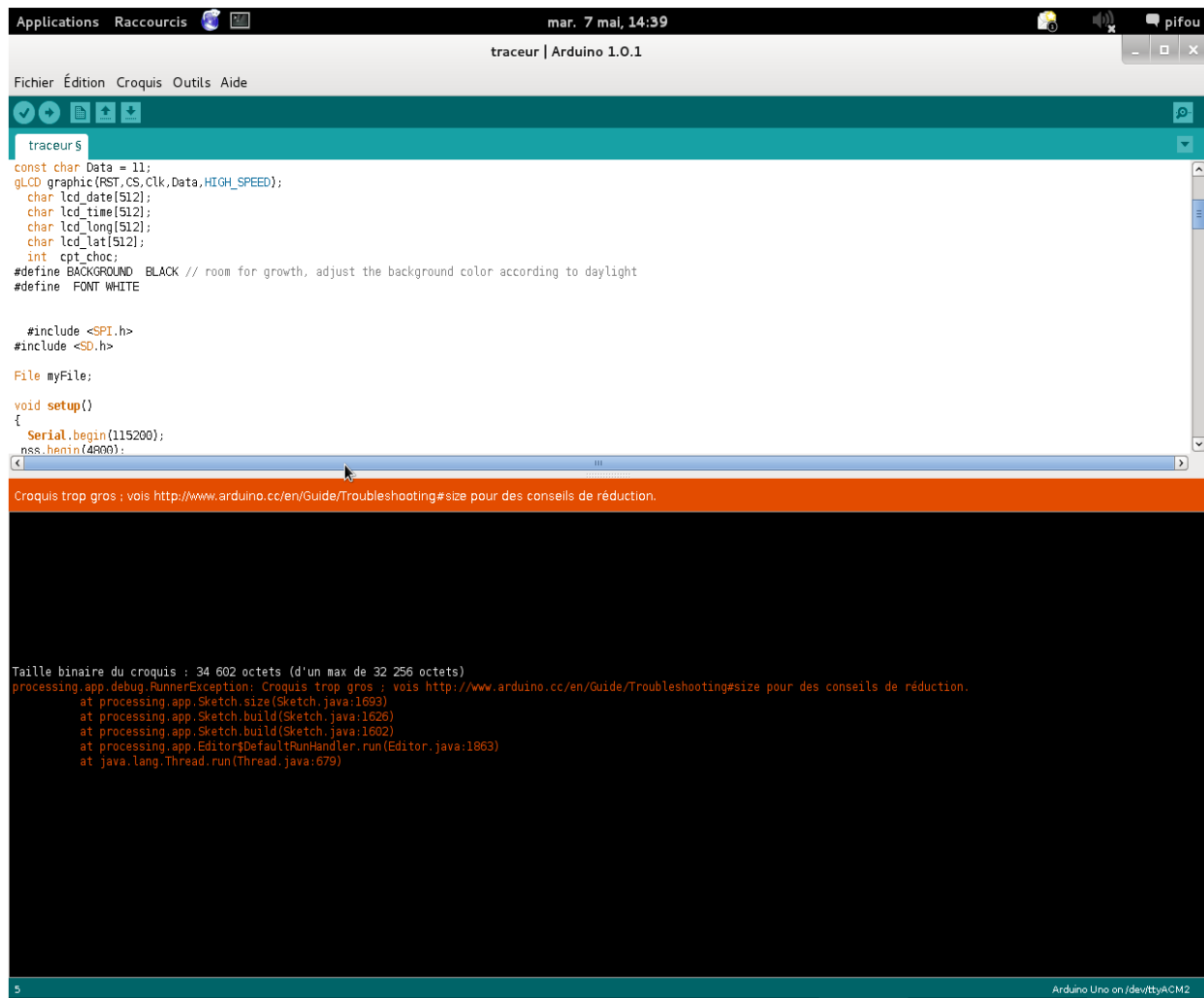
Malheureusement nous n'avons pas eu assez de temps pour concevoir entièrement la carte. En effet il nous a paru primordial de réaliser entièrement notre prototype basé sur arduino avant de concevoir notre version miniaturisé. Notre carte ne pouvant être imprimée qu'une fois, il fallait qu'elle puisse épouser toutes les contraintes que nous découvriions au fil de l'évolution de notre premier prototype.

Problèmes Rencontrés

Dans un premier temps, nous avons décelé des problèmes d'incompatibilité entre les bibliothèques «SoftwareSerial.h » et « wire.h » et nos soupçons ont été confirmés dans plusieurs blogs et forums.

Dans un second temps, nous avons eu un problème de dépassement de mémoire flash car notre programme était beaucoup trop lourd pour être chargé dans l'arduino UNO. Nous avons donc décidé d'utiliser l'arduino MEGA 2650 qui possède une mémoire assez importante, et plusieurs port série en hard.

Schéma montrant qu'on a bien un dépassement de mémoire :



```
Applications Raccourcis mar. 7 mai, 14:39 pifou
traceur | Arduino 1.0.1
Fichier Édition Croquis Outils Aide
traceur $
const char Data = 11;
glcd_graphic(RST,CS,Clk,Data,HIGH_SPEED);
char lcd_date[512];
char lcd_time[512];
char lcd_long[512];
char lcd_lat[512];
int cpt_choc;
#define BACKGROUND BLACK // room for growth, adjust the background color according to daylight
#define FONT WHITE

#include <SPI.h>
#include <SD.h>

File myFile;

void setup()
{
  Serial.begin(115200);
  nss.hemin(4880);
}

Croquis trop gros ; vois http://www.arduino.cc/en/Guide/Troubleshooting#size pour des conseils de réduction.

Taille binaire du croquis : 34 602 octets (d'un max de 32 256 octets)
processing.app.debug.RunnerException: Croquis trop gros ; vois http://www.arduino.cc/en/Guide/Troubleshooting#size pour des conseils de réduction.
at processing.app.Sketch.size(Sketch.java:1693)
at processing.app.Sketch.build(Sketch.java:1626)
at processing.app.Sketch.build(Sketch.java:1602)
at processing.app.Editor$DefaultRunHandler.run(Editor.java:1863)
at java.lang.Thread.run(Thread.java:679)

5 Arduino Uno on /dev/ttyACM2
```

Conclusion

Ce projet, très intéressant, a été pour nous un grand enrichissement technique et nous a permis de nous familiariser avec les travaux de recherches. Ces 10 semaines de projet nous ont confortés dans l'idée que nous nous faisons de la recherche. Nous avons appris à obtenir des résultats en partant de rien au départ.

Bien au-delà du travail que nous avons effectué, nous avons pu voir une vision globale de la recherche et des problèmes auxquels on est confrontés et comment les résoudre. Il nous a permis d'appliquer nos connaissances théoriques sur un projet concret. De plus ce projet nous semblait d'autant plus intéressant de part le fait qu'il tentait de résoudre un problème de la vie de tous les jours. Nous avons pris très à cœur la réalisation de notre projet et nous sommes plutôt content du travail rendu. Toutes fois nous aurions voulu avoir plus de temps pour terminer complètement ce projet afin de vous proposer une version plus compact et commercialisable du prototype que nous avons réalisé.

Ce projet a accrue notre motivation de devenir ingénieur, et pourquoi pas s'orienter vers une thèse qui pourra apporter beaucoup à la société d'aujourd'hui.

Annexes

Voici le programme final utilisé pour notre prototype :

```
#include <TinyGPS.h>

/* This sample code demonstrates the normal use of a TinyGPS object.
   It requires the use of SoftwareSerial, and assumes that you have a
   4800-baud serial GPS device hooked up on pins 3(rx) and 4(tx).
*/

TinyGPS gps;

static void gpstdump(TinyGPS &gps);
static bool feedgps();
static void print_float(float val, float invalid, int len, int prec);
static void print_int(unsigned long val, unsigned long invalid, int len);
static void print_date(TinyGPS &gps);
static void print_str(const char *str, int len);

#include <Wire.h>

#define DEVICE (0x53) //ADXL345 device address
#define TO_READ (6) //num of bytes we are going to read each time (two bytes for each axis)

byte buff[TO_READ] ; //6 bytes buffer for saving data read from the device
char acc[512];
#include <gLCD.h>
const char RST = 8;
const char CS = 9;
const char Clk = 13;
const char Data = 11;
gLCD graphic(RST,CS,Clk,Data,HIGH_SPEED);
char lcd_date[512];
char lcd_time[512];
char lcd_long[512];
char lcd_lat[512];
int cpt_choc;
#define BACKGROUND BLACK // room for growth, adjust the background color according to daylight
#define FONT WHITE

#include <SPI.h>
#include <SD.h>

File myFile;

void setup()
```

```

{
Serial.begin(115200);
Serial3.begin(4800);
Wire.begin(); // join i2c bus (address optional for master)
pinMode(10, OUTPUT);

//Turning on the ADXL345
writeTo(DEVICE, 0x2D, 0);
writeTo(DEVICE, 0x2D, 16);
writeTo(DEVICE, 0x2D, 8);
cpt_choc=0;
graphic.begin(0,0,0,PHILLIPS_0);
graphic.setBackColour(15,15,15);
graphic.setForeColour(0,0,0);
graphic.setFont(Normal_SolidBG);
graphic.setCoordinate(5,0);
graphic.print("TRACEUR DE CHOC");
graphic.setCoordinate(5,10);
graphic.print(" BY HOSSIE & KEMAJOU");

Serial.print("Traceur de choc "); Serial.println(TinyGPS::library_version());
Serial.println("by HOSSIE & KEMAJOU IMA4 SC 2012/2013");
Serial.println();
Serial.print("Sizeof(gpsobject) = "); Serial.println(sizeof(TinyGPS));
Serial.println();
Serial.println("Sats HDOP Latitude Longitude Fix Date Time Date Alt Course Speed Card
Distance Course Card Chars Sentences Checksum acceleration");
Serial.println(" (deg) (deg) Age Age (m) --- from GPS ---- ---- to PARIS ----
RX RX Fail x y z");
Serial.println("-----");
-----");
graphic.setCoordinate(5,30);
graphic.print("DATE:");
graphic.setCoordinate(5,40);
graphic.print("TIME:");
graphic.setCoordinate(5,50);
graphic.print("LONGITUDE:");
graphic.setCoordinate(5,60);
graphic.print("LATITUDE:");
graphic.setCoordinate(5,70);
graphic.print("ACC");
graphic.setCoordinate(5,80);
graphic.print("NBCHOC");
graphic.setCoordinate(5,90);
if (!SD.begin(53)) {
graphic.print("initialization failed!");
}
graphic.print("initialization done.");
myFile = SD.open("logger.txt", FILE_WRITE);

if (myFile) {

```

```
myFile.println("TRACEUR DE CHOC");
    // close the file:
myFile.close();

} else {
    // if the file didn't open, print an error:
    graphic.println(" error opening file");
}
}

void loop()
{
    bool newdata = false;
    unsigned long start = millis();

    // Every second we print an update
    while (millis() - start < 1000)
    {
        if (feedgps())
            newdata = true;
    }

    gpsdump(gps);
    int regAddress = 0x32; //first axis-acceleration-data register on the ADXL345
    int x, y, z;

    readFrom(DEVICE, regAddress, TO_READ, buff); //read the acceleration data from the ADXL345

    //each axis reading comes in 10 bit resolution, ie 2 bytes. Least Significant Byte first!!
    //thus we are converting both bytes in to one int
    x = (((int)buff[1]) << 8) | buff[0];
    y = (((int)buff[3])<< 8) | buff[2];
    z = (((int)buff[5]) << 8) | buff[4];

    //we send the x y z values as a string to the serial port
    sprintf(acc, "%02d\t%02d\t%02d", x, y, z);
    Serial.println(acc);

    graphic.setCoordinate(70,30);
    graphic.print(lcd_date);

    graphic.setCoordinate(70,40);
    graphic.print(lcd_time);

    graphic.setCoordinate(70,50);
    graphic.print(lcd_long);

    graphic.setCoordinate(70,60);
    graphic.print(lcd_lat);

    graphic.setCoordinate(60,70);
    graphic.print(acc);
```

```

graphic.setCoordinate(60,80);
graphic.print(cpt_choc);

if ((abs(x)>300)||abs(y)>300)||abs(z)>300)
{
  cpt_choc++;
  graphic.setCoordinate(40,100);
  graphic.print("#####CHOC#####");
  delay(1000);
  graphic.setForeground(15,15,15);
  graphic.print(" ");
  graphic.setForeground(0,0,0);

  myFile = SD.open("logger.txt", FILE_WRITE);
  if (myFile) {

    Serial.println("#####CHOC#####");
    myFile.println("-----");
    myFile.println("#####CHOC#####");
    myFile.print("DATE: ");
    myFile.println lcd_date;
    myFile.print("TIME: ");
    myFile.println lcd_time;
    myFile.print("LONGITUDE: ");
    myFile.println lcd_long;
    myFile.print("LATITUDE: ");
    myFile.println lcd_lat;
    myFile.println("-----");
    // read from the file until there's nothing else in it:

    // close the file:
    myFile.close();
  } else {
    graphic.setCoordinate(5,80);
    graphic.println("error opening test.txt");
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }

}

}

static void gpsdump(TinyGPS &gps)
{
  float flat, flon;
  unsigned long age, date, time, chars = 0;
  unsigned short sentences = 0, failed = 0;
  static const float PARIS_LAT = 48.85661, PARIS_LON = 2.35222;

  print_int(gps.satellites(), TinyGPS::GPS_INVALID_SATELLITES, 5);
  print_int(gps.hdop(), TinyGPS::GPS_INVALID_HDOP, 5);

```



```

gps.f_get_position(&flat, &flon, &age);
print_float(flat, TinyGPS::GPS_INVALID_F_ANGLE, 9, 5);
print_float(flou, TinyGPS::GPS_INVALID_F_ANGLE, 10, 5);
print_int(age, TinyGPS::GPS_INVALID_AGE, 5);
dtostrf(flou, 7, 6, lcd_long);
dtostrf(flat, 7, 6, lcd_lat);
//sprintf(lcd_long, "%d", flon*100000);
// sprintf(lcd_lat, "%d", flat*100000);

print_date(gps);

print_float(gps.f_altitude(), TinyGPS::GPS_INVALID_F_ALTITUDE, 8, 2);
print_float(gps.f_course(), TinyGPS::GPS_INVALID_F_ANGLE, 7, 2);
print_float(gps.f_speed_kmph(), TinyGPS::GPS_INVALID_F_SPEED, 6, 2);
print_str(gps.f_course() == TinyGPS::GPS_INVALID_F_ANGLE ? "**** " :
TinyGPS::cardinal(gps.f_course()), 6);
print_int(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0UL : (unsigned
long)TinyGPS::distance_between(flat, flon, PARIS_LAT, PARIS_LON) / 1000, 0xFFFFFFFF, 9);
print_float(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : TinyGPS::course_to(flat, flon,
51.508131, -0.128002), TinyGPS::GPS_INVALID_F_ANGLE, 7, 2);
print_str(flat == TinyGPS::GPS_INVALID_F_ANGLE ? "**** " :
TinyGPS::cardinal(TinyGPS::course_to(flat, flon, PARIS_LAT, PARIS_LON)), 6);

gps.stats(&chars, &sentences, &failed);
print_int(chars, 0xFFFFFFFF, 6);
print_int(sentences, 0xFFFFFFFF, 10);
print_int(failed, 0xFFFFFFFF, 9);
// Serial.println();
}

static void print_int(unsigned long val, unsigned long invalid, int len)
{
char sz[32];
if (val == invalid)
strcpy(sz, "*****");
else
sprintf(sz, "%ld", val);
sz[len] = 0;
for (int i=strlen(sz); i<len; ++i)
sz[i] = ' ';
if (len > 0)
sz[len-1] = ' ';
Serial.print(sz);
feedgps();
}

static void print_float(float val, float invalid, int len, int prec)
{
char sz[32];
if (val == invalid)
{
strcpy(sz, "*****");
sz[len] = 0;

```

```

    if (len > 0)
        sz[len-1] = ' ';
    for (int i=7; i<len; ++i)
        sz[i] = ' ';
    Serial.print(sz);
}
else
{
    Serial.print(val, prec);
    int vi = abs((int)val);
    int flen = prec + (val < 0.0 ? 2 : 1);
    flen += vi >= 1000 ? 4 : vi >= 100 ? 3 : vi >= 10 ? 2 : 1;
    for (int i=flen; i<len; ++i)
        Serial.print(" ");
}
feedgps();
}

static void print_date(TinyGPS &gps)
{
    int year;
    byte month, day, hour, minute, second, hundredths;
    unsigned long age;
    gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);
    if (age == TinyGPS::GPS_INVALID_AGE)
        Serial.print("*****      ***** ");
    else
    {
        char sz[32];
        sprintf(sz, "%02d/%02d/%02d %02d:%02d:%02d  ",
            month, day, year, hour, minute, second);
        sprintf(lcd_time, "%02d:%02d:%02d", hour, minute, second);
        sprintf(lcd_date, "%02d/%02d/%02d", month, day, year, hour);
        Serial.print(sz);
    }
    print_int(age, TinyGPS::GPS_INVALID_AGE, 5);
    feedgps();
}

static void print_str(const char *str, int len)
{
    int slen = strlen(str);
    for (int i=0; i<len; ++i)
        Serial.print(i<slen ? str[i] : ' ');
    feedgps();
}

static bool feedgps()
{
    while (Serial3.available())
    {
        if (gps.encode(Serial3.read()))
            return true;
    }
}

```

```
}
return false;
}
void writeTo(int device, byte address, byte val) {
    Wire.beginTransmission(device); //start transmission to device
    Wire.write(address); // send register address
    Wire.write(val); // send value to write
    Wire.endTransmission(); //end transmission
}
void readFrom(int device, byte address, int num, byte buff[]) {
    Wire.beginTransmission(device); //start transmission to device
    Wire.write(address); //sends address to read from
    Wire.endTransmission(); //end transmission

    Wire.beginTransmission(device); //start transmission to device (initiate again)
    Wire.requestFrom(device, num); // request 6 bytes from device

    int i = 0;
    while(Wire.available()) //device may send less than requested (abnormal)
    {
        buff[i] = Wire.read(); // receive a byte
        i++;
    }
    Wire.endTransmission(); //end transmission
}
```

Sources

<https://www.sparkfun.com/datasheets/GPS/NMEA%20Reference%20Manual1.pdf>
<http://blog.bricogeek.com/noticias/tutoriales/tutorial-arduino-gps-logger-con-em406a-gps-shield-y-microsd-shield/>
<http://glaikit.org/2011/02/11/garmin-vs-arduino-gps-dance-off/>
http://flickrhivemind.net/flickr_hvmnd.cgi?method=GET&sorting=Interestingness&page=2&photo_type=250&noform=t&search_domain=Tags&photo_number=50&tag_mode=all&sort=Date%20Posted,%20new%20first&textinput=arduino,gps&origininput=arduino,gps&search_type=Tags
<http://hackaday.com/2012/11/23/arduino-data-logger-maps-out-the-potholes-on-your-morning-commute/>
<http://www.instructables.com/id/Track-your-route-using-arduino-microSD-card-shi/>
<http://skyduino.wordpress.com/2012/04/03/comparatif-des-differentes-cartes-arduino-et-des-cartes-compatible-arduino/>
<http://snootlab.com/adafruit/264-module-gps-em-406a.html>
<http://forum.arduino.cc/index.php?topic=79047.0>
http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.MaterielCapteurModuleGPSEM406SiRFIIIFiche
<http://bildr.org/2011/06/em406a-gps-arduino/>
http://www.electfreaks.com/wiki/index.php?title=File:BK_Accelerometer_ADXL345_04.jpg
http://www.evola.fr/product_info.php/shield-carte-microsd-arduino-p-68
<http://www.lextronic.fr/P5912-module-carte-sd-pour-arduino.html>