Université de Lille

POLYTECH LILLE

Rapport de projet SafeWatch

IMA4 SC

Réalisé par : NAANAA Oumaima

Encadré par :
Mr. REDON Xavier

Année universitaire 2017-2018

Sommaire

Introduction

Des personnes en difficultés peuvent se trouver seules à l'extérieur et ont la gêne de demander l'aide à des étrangers . La SafeWatch une montre simple d'utilisation, leurs permet de rentrer chez eux au plus vite et d'appeler leurs proches en cas de besoin.

Ce rapport décrit le prototype ainsi que la réalisation de cette montre en détaillant les différentes parties logicielles et matériels relatives .

Dans une première partie , je présenterai le projet et le cahier de charge défini préalablement ainsi que le choix de matériel pour un prototype puis pour une réalisation .

La deuxième partie sera consacrée pour le prototype puis une troisième partie pour la réalisation . Dans les deux parties je détaillerai les tâches réalisées pour la géolocalisation et la communication GSM , j'expliquerai les codes effectuant ses fonctions ainsi que le difficultés rencontrées . Je citerai par la suite les solutions mises en place .

1.Présentation du projet

1.1 Objectif et cahier de charge :

La SafeWatch est une montre connectée simple d'utilisation permettant de réorienter les personnes en difficultés. Elle permet également de demander l'aide à un référent en lui envoyant un SMS contenant la localisation GPS ou de l'appeler directement .

Un écran affiche des flèches d'orientation une fois un des deux boutons poussoir est appuyé. Les flèches aident les personnes en difficulté de se déplacer en se référant uniquement aux sens affichés , ce qui est plus simple d'utilisation qu'un GPS de téléphone par exemple .

La safeWatch contient donc un module GPS qui lui permet la localisation et un module GSM assurant l'envoi des SMS et des appels. Sans oublier les deux boutons poussoirs pour choisir entre une orientation "indépendante" ou une demande d'aide d'un référent.

1.2 Choix du matériel:

Ce projet était réalisé en premier lieu par un matériel "encombrant" fourni par Mr.REDON,afin de commencer par un prototype , avant de commander un nouveau plus "petit" pour que l'ensemble ressemble plus à une montre .

Premier matériel utilisé :

- Arduino UNO et 3 boutons poussoirs.
- GPS Adafruit v3.
- bouclier GSM Arduino.
- Affichage d'horloge et de l'orientation GPS vers domicile : 2.8" TFT LCD shield w/Touchscreen

Etant donné que l'objectif du 2ème matériel est de rendre la montre la plus petite possible , l'idée était donc d'utiliser un circuit regroupant GPS et GSM ainsi qu'un écran disposant d'un microprocesseur :

- Adafruit FONA 808 Cellular + GPS Breakout
- OLED Display de MicroView + USB Programmer

Auxquels s'ajoutent :

- Antenne GPS

- Antenne GSM
- Batterie :1200mAh sized Lithium ion/polymer battery

2. Prototype SafeWatch

Dans ce chapitre , je décrirai les différentes étapes effectuées afin de réaliser les 4 fonctions principales de la montre : localisation, orientation , SMS et appel vocal ; que cela soit avec le matériels de prototype ou celui de la réalisation .

2.1 Géolocalisation GPS:

Je détaille dans cette partie la localisation GPS pour le prototype ainsi que pour la materiels de réalisation :

Prototype : 1 er matériel

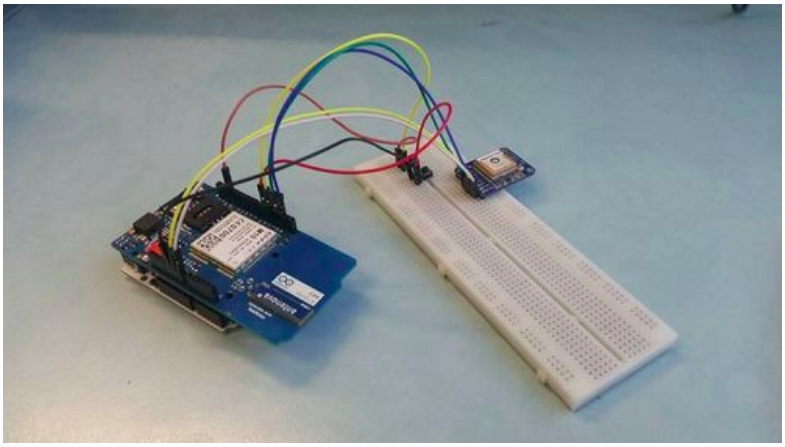

Image 1 : montage prototype GSM+GPS

Afin d'obtenir une localisation , on s'intéresse à la latitude et la longitude . Le GPS nous ne donne pas que ces deux informations, on obtient également le temps,la date, la vitesse et bien d'autres paramètres. Ces informations sont fournies sous formes de "phrase" , qu'on appelle les "NMEA sentences".

NMEA Sentences :

Un GPS nous permet d'avoir plusieurs types de NMEA sentence , on s'intéressera qu'à *$GPRMC* et *$GPGGA* qui donnent la latitude et la longitude l'altitude ainsi que l'état du FIX .

FIX : est la position obtenue par 3 satellites (2D fix) ou 4 satellites (3D fix) . Si la valeur du fix est invalide cela prouve que le GPS n'arrive pas à communiquer avec les satellites .

$GPRMC :The recommended minimum sentence C
Exemple :

*$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A*

$GPRMC,temps,etat_fix,**latitude**,point_cardinal,**longitude**,point_cardinal,vitesse_gps,track_angle,date,variation_magnétique,checksum

*etat_fix = **A**:Active ou **V**:void

$GPGGA: Global Positioning System Fix Data

Exemple :

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

$GPGGA,temps,latitude,point_cardinal,longitude,point_cardinal,qualité_fix,nb_sattelites,GDOP,**altitude** ,hauteur_geoid,,checksum
*qualité_fix**: 0** = invalid / **1** = GPS fix

Test & obtention NMEA :

Pour récupérer les NMEA sentences , je branche les pins TX et RX du GPS au pin 0 et 1 de l'Arduino . Cela me permet d'établir une communication série avec l'ordinateur en passant par Arduino , le GPS envoi la data sans programmer l'Arduino :

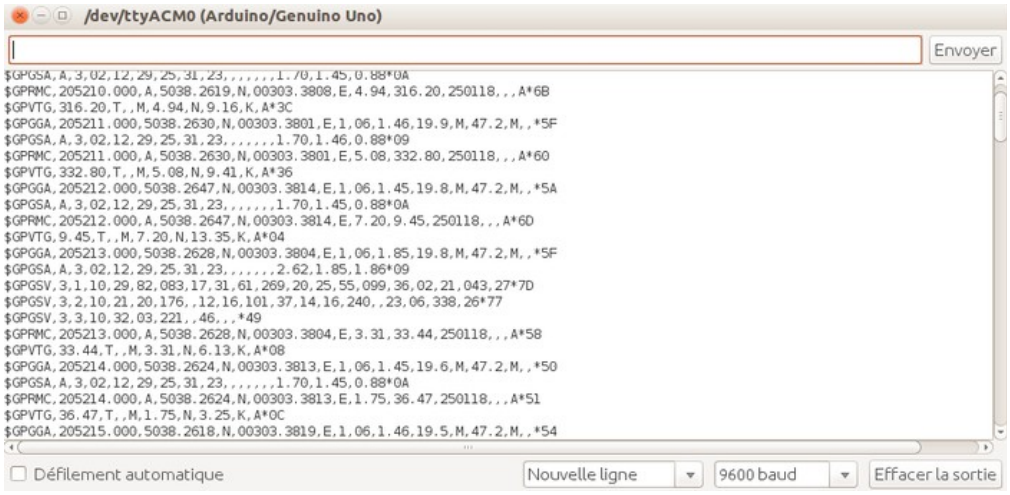

Image2 : NMEA sentences

Cela me donne beaucoup d'information , il a fallu donc programmer l'arduino afin de filtrer l'ensemble et extraire le nécessaire . Ce code (Code semaine 1 en annexes) permet donc d'avoir le *temps ,la date,la localisation (latitude et longitude), vitesse, angle ,* et le *nombre de satellites* en communication.

Remarque :
Parfois je n'arrivais pas à avoir une bonne réception , et cela était résolu en laissant du temps au GPS . La réception n'était pas instantanée .

Image 3: Informations nécessaire extraites des NMEA Sentences

Pour vérifier ces coordonnées , vous pouvez utiliser le lien suivant :
"http://maps.google.com/?q=<latitude en degrés>,<longitude en degrés>"
http://maps.google.com/?q=50.6376,3.0564
Ce format de lien sera utilisé ensuite pour l'envoi de la position par SMS.

2.2Envoi SMS :

Selon le cahier de charge , la montre envoie un SMS contenant un lien Google maps qui affiche la position de la montre . Cette partie est fonctionnelle aussi bien pour le prototype que pour la réalisation .
En rassemblant l'arduino UNO et le shield GSM , on obtient le "montage GSM". Le code de cette partie consiste à envoyer un SMS une fois le bouton poussoir est appuyé (Vidéo démonstrative sur le wiki) . Le code de cette partie est celui de la semaine 2 . Ce code utilise les bibliothèques Adafruit_GPS.h et GSM.h . La fonction *<objet>*.print(message) prend en paramètre un *String* : il a fallu donc convertir la latitude et la longitude qui sont des *float* en utilisant la fonction String():

String latitude = String(GPS.latitudeDegrees);

```
String longitude= String(GPS.longitudeDegrees);
sms.print("lien google maps: http://maps.google.com/?q="+latitude+","+longitude);
```

2.3 Appel au référent :

L'appel sera effectué une fois le bouton 1 est appuyé. Si on veut raccrocher on appuie encore une fois sur le même bouton , ce test ne sera pas pris en charge pour la réalisation projet . Il est plus sécurisé de raccrocher du coté de référent .Le code est celui de la semaine 1 et utilise la bibliothèque GSM.h également . L'appel et la fin d'appel ne sont pas contrôlé par le bouton uniquement mais aussi par l'état d'appel :

```
/* Pas d'appel en cours=> appel effectué */
if (buttonState1 == HIGH && (appel.getvoiceCallStatus()==IDLE_CALL)) {
  (appel.voiceCall(remoteNum))}

/*Appel en cours => raccrocher */
else if (buttonState1 ==HIGH && (appel.getvoiceCallStatus()==TALKING)){
appel.hangCall()  }
```

La dernière mise à jour de ce code ne permettait pas de raccrocher du côté de la montre , mais cela n'impacte pas le fonctionnement imposé par le cahier de charge . Je ne me suis pas tardé sur ce code , et je me suis concentrée sur la programmation du matériel de réalisation .

2.4 Orientation par flèches :

Calcul d'itinéraire :

La Terre forme quatre quadrants : NORD-EST , NORD-OUEST , SUD-EST , SUD-OUEST . La France fait partie du NORD-EST , plus précisément entre 42° et 52° en latitude , 2° et 9° en longitude .

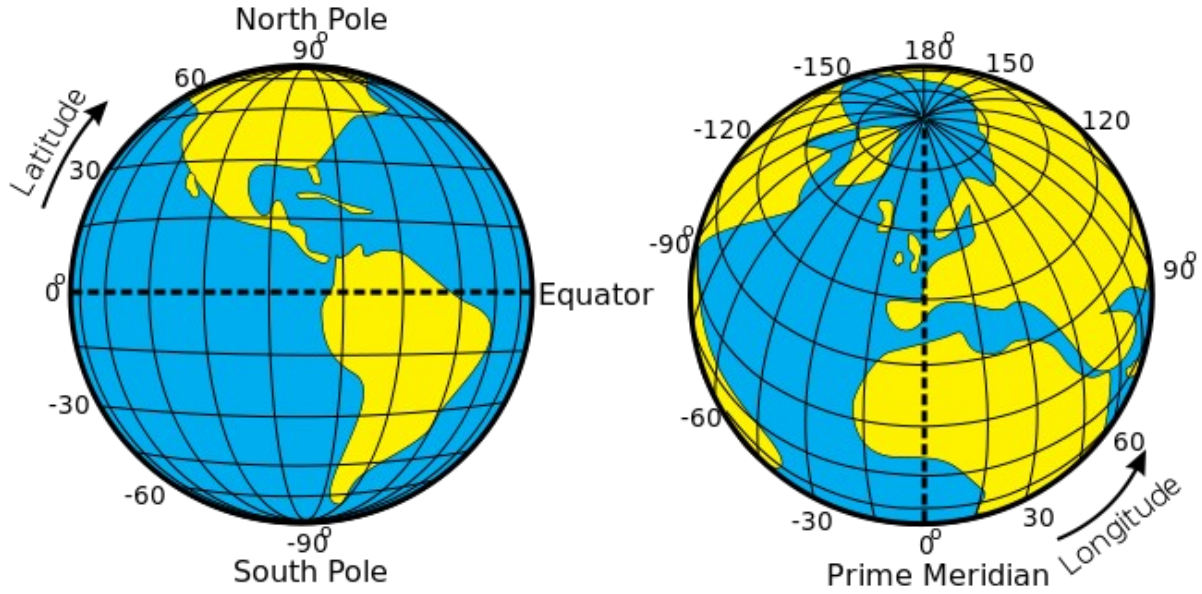

Image 4 : Représentation de la latitude et la longitude

Supposant les adresses suivantes :

Domicile : Résidence Eiffel 51 rue de Ticléni Villeneuve d'ascq

Position : Polytech Lille Avenue Paul Langevin, 59655 Villeneuve-d'Ascq

Adresses	Latitude	Longitude
Position	50.6076754	3.1363522999999986
Domicile	50.61108650000001	3.1490148000000318

L'orientation des flèches se fera selon la différence d'angle (latitude et longitude) de la position par rapport au domicile . Sachant qu'on se situe au quadrant NORD-EST :

- Si (latitude_Domicile - latitude_Position > 0)

alors clignoter la flèche "Devant" sinon clignoter la flèche "Derrière"

- Si (longitude_Domicile - longitude_Position > 0)

alors clignoter la flèche "Droite" sinon clignoter la flèche " Gauche"

Dans notre cas , on devra avoir les flèches "Droite" et "Devant" qui clignotent .

Vérification sur Google Maps :

Image 5: Vérification du calcul d'itinéraire par Google Maps

Affichage des flèches par le prototype :

La résolution du LCD est de 240x320 .
Les quatres fleches partagent l'ecran , elles seront affichées toutes à la fois . La flèche concerné va clignoter .
L'interet d'afficher les mots "droite" , "gauche" est de ne pas confondre les sens des flèches si on tourne l'écran. Normalement la connaissance d'un sens nous permettera de reconnaitre les autres .

Travail sur papier :

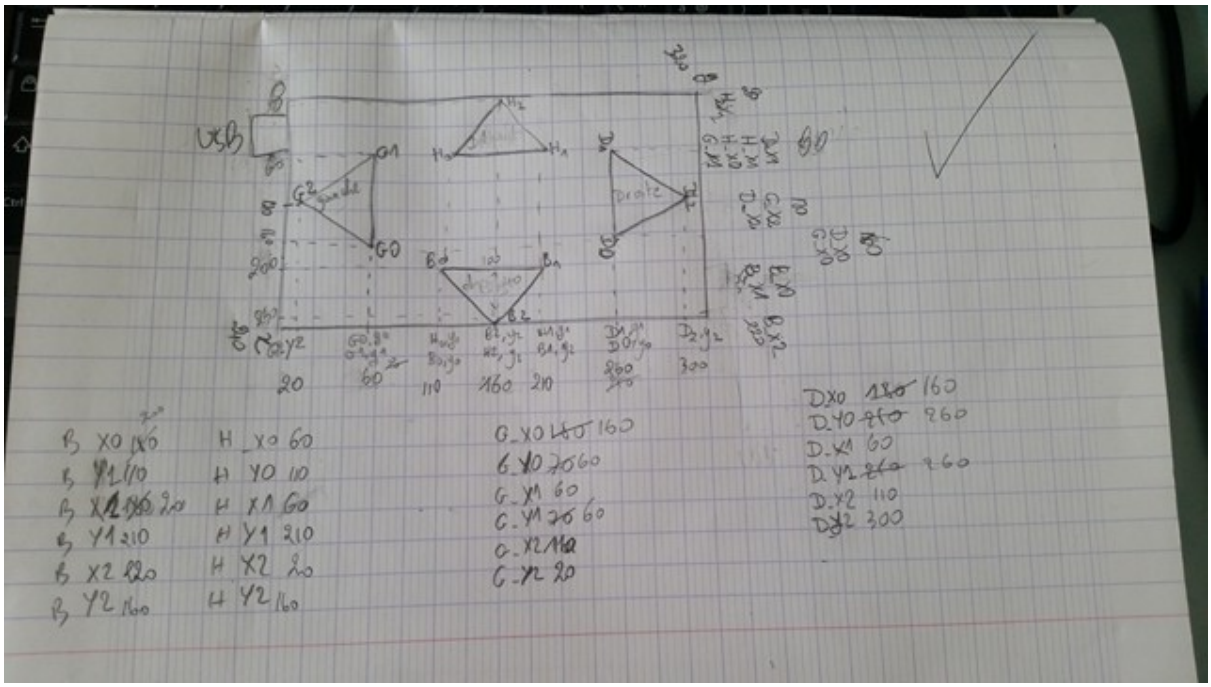

Image 6: Flèches d'orientation prototype sur papier

Codes et résultats :

Le code de la semaine 4 permet de tracer 4 triangles verts sur un fond rose , marque "droite" et "gauche" en noir sur les triangles correspondants .

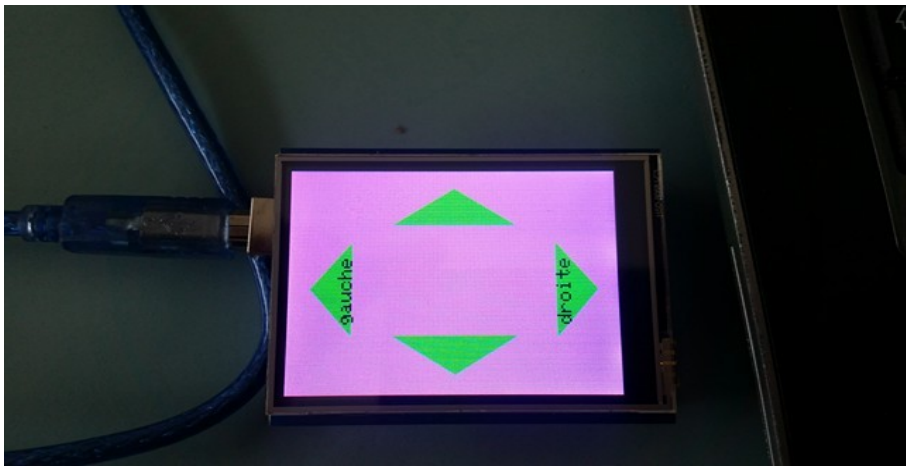

Image 7: Flèches prototype

Le code "semaine5" fait clignoter la bonne flèche selon les conditions vues en calcul d'itinéraire . Ce code n'est pas testé sur le matériel prototype car l'LCD ne permet l'accès au pin d'Arduino afin de la connecter avec le GPS . L'algorithme sera vérifié par l'afficheur MicroViewer .

3. Réalisation de la SafeWatch :

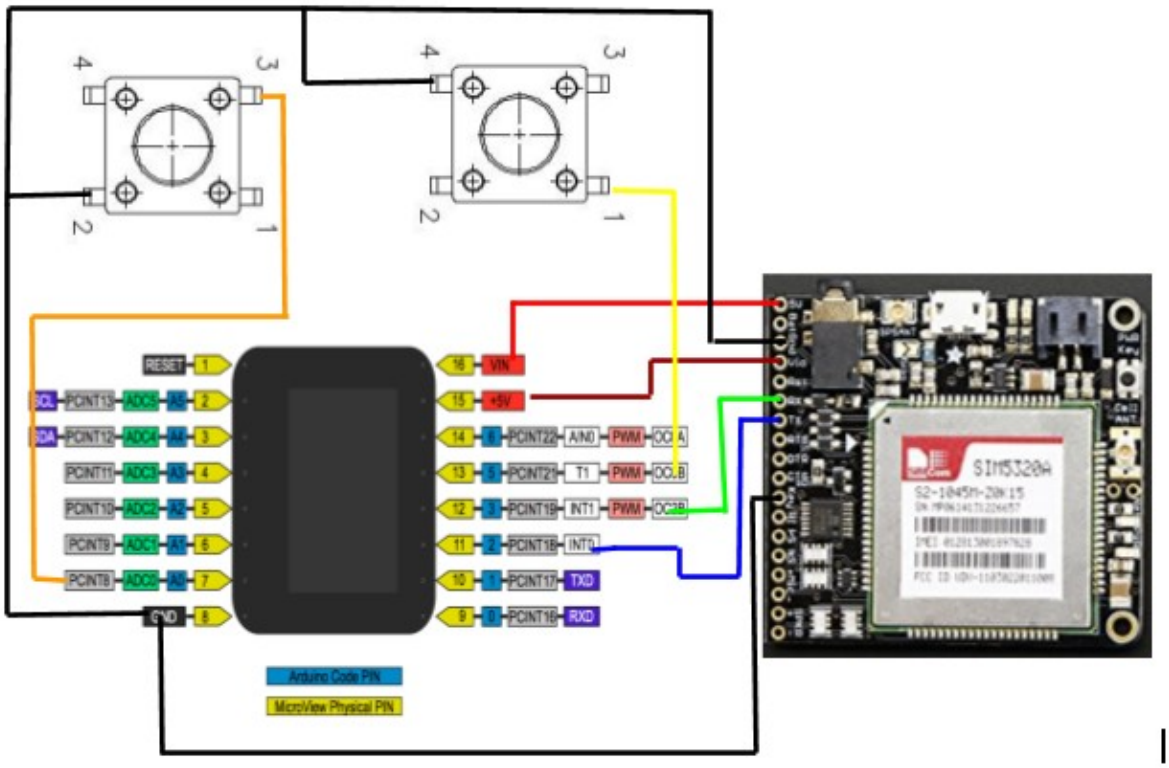

Image 8: Montage matériel de réalisation

3.1 Géolocalisation GPS

L'idée est toujours la même , mais cette fois j'ai eu l'avantage d'avoir le code source du circuit *FONA 3G*. En utilisant les fonctions GPS du code source *boolean enableGPS(boolean onoff);* et *boolean getGPS(float *lat, float *lon, float *speed_kph=0, float *heading=0, float *altitude=0);* , j'ai pu communiquer avec le module *SIMCOM5320E* via des commandes AT. ex: *AT+CGPS* qui donne l'état du gps on/off *; AT+CGPSINFO* qui renvoi l'information fournie par GPS .

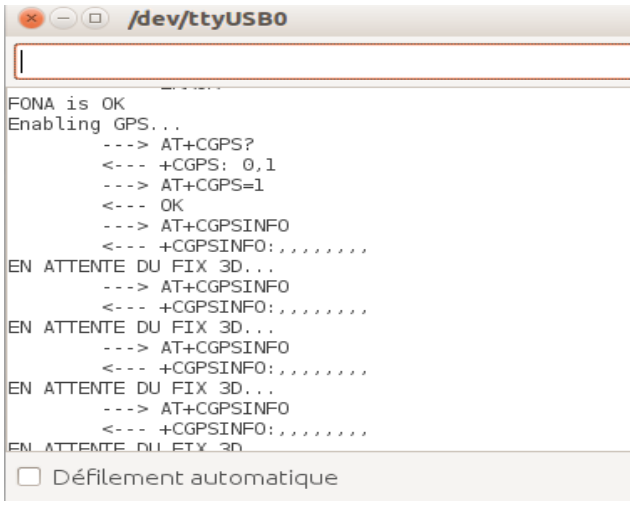

Image 9: résultat gps (code semaine 9)

Dans ce cas , l'obtention du fix n'était pas possible malgré le fait d'essayer toutes les solutions possibles .

En partant de l'idée d'avoir un problème software, j'ai opté pour une **géolocalisation GSM** par **cell-ID** ou par **Triangulation**, mais celle-ci n'était pas "faisable" par le circuit FONA 3G(Source : datasheet p48). Une géolocalisation par internet aurait été possible , vu la particularité de ce circuit par rapport aux autres de la même familles d'où son choix. Le principe est d'envoyer des requêtes à deux serveurs Google API : http://www.google.com/loc/json et http://www.google.com/glm/mmap afin d'obtenir un point géographique et un rayon précis à partir des données des antennes-relais détectés par FONA 3G. Ces deux serveurs ne sont plus accessibles et leur remplacement par d'autres serveurs est impossible car l'accès se fait directement par la commande AT+CASSISTLOC .

Finalement le problème est hardware et non software , la raison est le type d'antenne . Le circuit est conçu pour utiliser une antenne passive , l'antenne que j'utilise et bel et bien une antenne passive mais toujours pas de bonne réception du FIX . Après avoir comparé les schémas d'antenne dans hardware design du SIMCOM et celui de FONA 3G d'Adafruit fournis par la datasheet j'ai remarqué que SIMCOM exige deux self L1 et L2 pour une bonne sensibilité pourtant ces deux selfs ne figurent pas dans le schéma de FONA 3G :

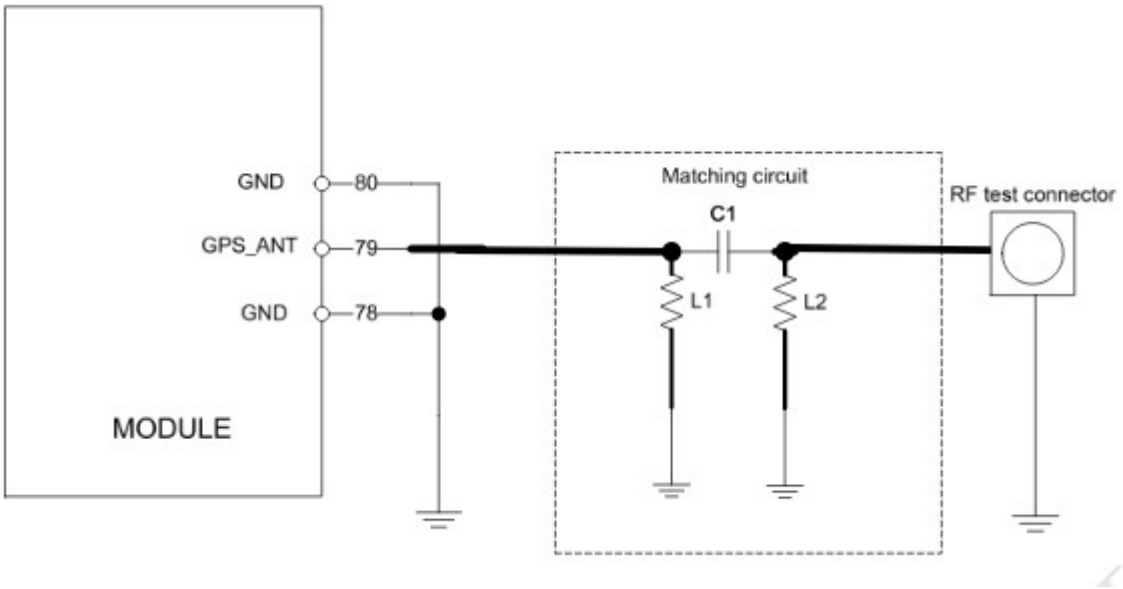

Figure 1 : Circuit de l'antenne GPS passive SIMCOM

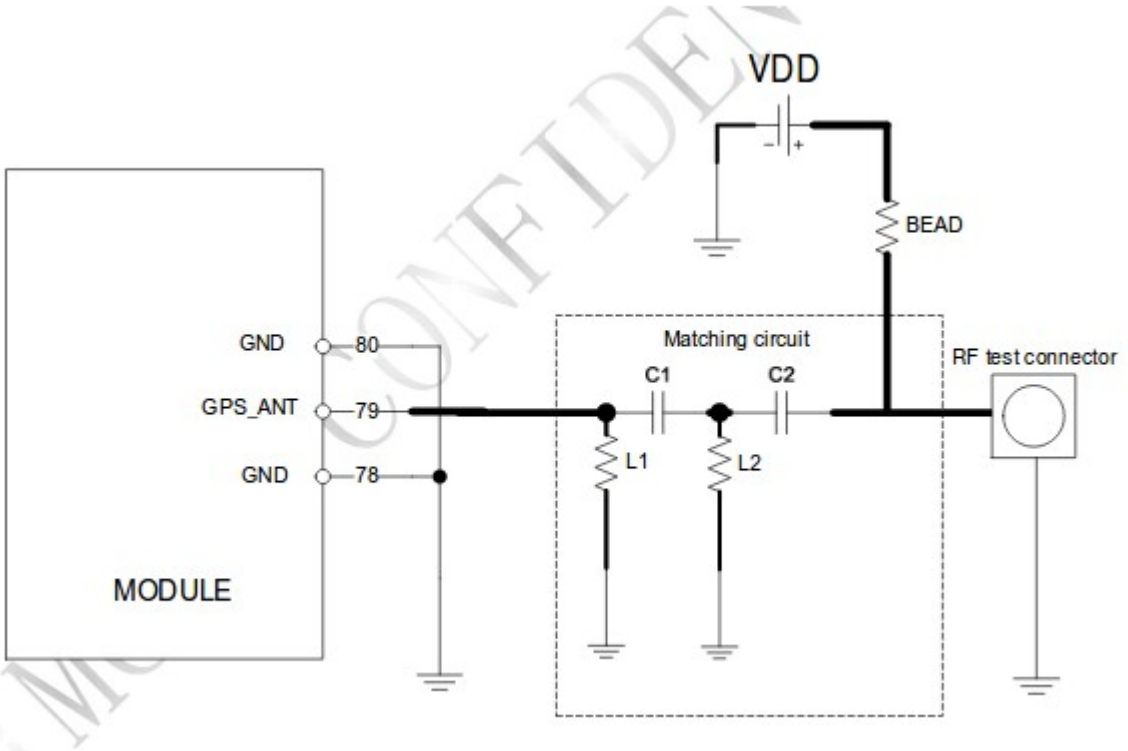

Figure 2 : Circuit de l'antenne GPS active SIMCOM

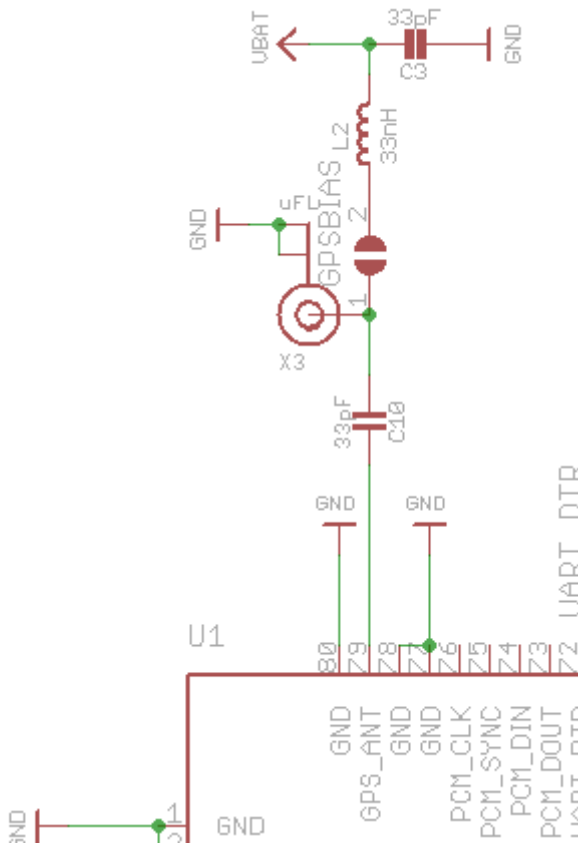

Figure 3: Circuit de l'antenne GPS Adafruit

Sachant que Adafruit a mis un bias dans son circuit pour offrir le choix entre une antenne passive et une antenne active . La solution proposé par Adafruit et souder ce bias et d'utiliser une antenne active . L'utilisation de ce type d'antenne est "impossible" pour notre type de projet , vu ses dimensions .

Image 10: Antenne GPS active

Il est aussi recommandé d'utiliser un LNA (Low-Noise Amplifier) pour "renforcer" le signal, vu que l'avantage d'une antenne active par rapport à une antenne passive est son amplificateur LNA intégré . Mais cela sera encore plus encombrant .

Le code de la semaine6_2 vérifie l'orientation en utilisant le GPS du prototype

(Adafruit GPS).

3.2 Envoi du SMS :

Le principe est toujours le même , le code utilisé est celui de la semaine 12. La particularité par rapport au code SMS du prototype est qu'il n'envoie le SMS que si le GPS arrive à obtenir ses coordonnées . Dans l'écriture de ce code , il a fallu effectuer une concaténation du message à envoyer contrairement au cas du prototype:

```
char  Slatitude[20];
char Slongitude[20];
/*Conversion de la latitude et  la longitudeen String*/

dtostrf(latitude,8,5,Slatitude);
dtostrf(longitude,8,5,Slongitude);// conversion de en String

/*ajouter le lien au message en premier puis les coordonnées */

strcpy(message,"lien google maps: http://maps.google.com/?q=");
strcat(message,Slatitude);
strcat(message,",");
strcat(message,Slongitude);
puts(message);
Serial.println(message);
fona.sendSMS(num,message);
```

3.3 Appel au référent :

Contrairement au shield GSM , le circuit FONA 3G offre la possibilité d'une communication vocale par des écouteurs avec un micro reliés au Headset jack du circuit . Vu que j'ai deux boutons , l'un est pour l'orientation que nous verrons par la suite et l'autre que j'utilise pour l'envoi du SMS qui permettra également d'appeler le référent mais après une attente de 5 secondes de l'envoi du SMS . Le code des semaines 10_11 utilise la bibliothèque Adafruit_FONA.h et précisément la fonction *boolean callPhone(char* num)*

```
/*Extrait du code */
void makingCall(){
  char num[20]="0753896813";
  fona.callPhone(num) ;}
```

3.4Orientation par flèches :

Les dimensions de l'afficheur Microview ne permet pas d'afficher les flèches à la fois. Il affichera une flèche à la fois . Pour tester l'affichage de toutes les flèches , le code "semaine 6_1" flèche par flèche chaque seconde .
Pour tracer un flèche , j'ai "dessiné" un triangle collé à un rectangle .

L'orientation est réalisé par le code "semaine6_1" se basant sur le même algorithme du calcul d'itinéraire . J'ai utilisé le GPS du matériel du prototype qui donne des coordonnées valides .

Résultat :

Image 11: Flèche sur Microview

Pour changer le sens de la flèche il fallait se déplacer . Je n'ai pas prévu une alimentation pour ce type de montage pour qu'il soit "portable".

3.5 Programme global :

Le "code_final" affiche une horloge analogique , permet la gélocalisation , l'envoi SMS , l'appel au référent en appuyant sur le bouton 1 ainsi que l'orientation par flèches en appuyant sur le bouton 2 .

3.6 Réalisation de la carte :

L'intérêt de la carte est de rassembler l'ensemble du matériel . Elle n'a aucun impacte sur le montage réalisé sur la board et par conséquent ne demande pas de composants électroniques autres que le matériel utilisé . Elle contien 2 Header de 8 pour l'afficheur et un Header de 18 pour le circuit FONA et 2 boutons poussoirs .

Schematic :

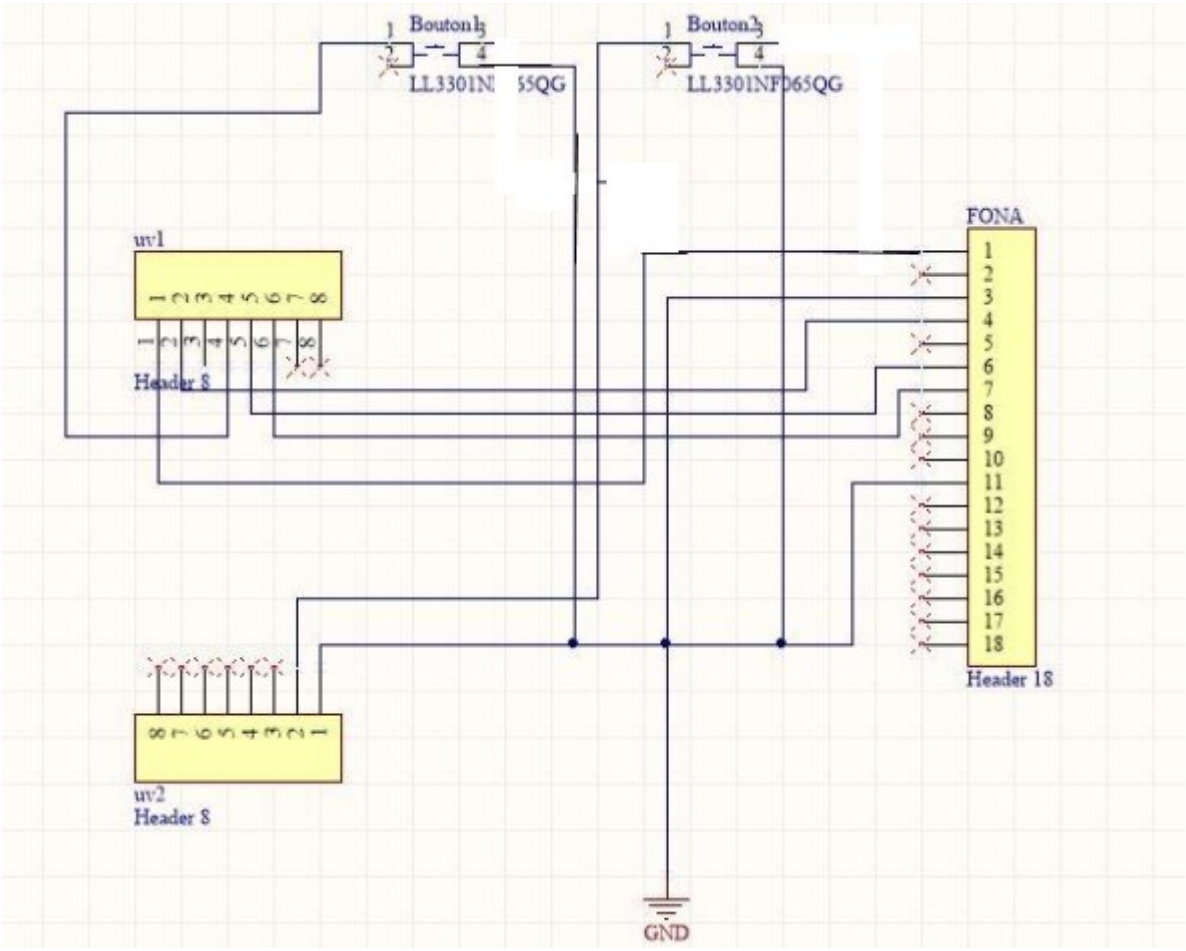

Image 12: Montage de la carte

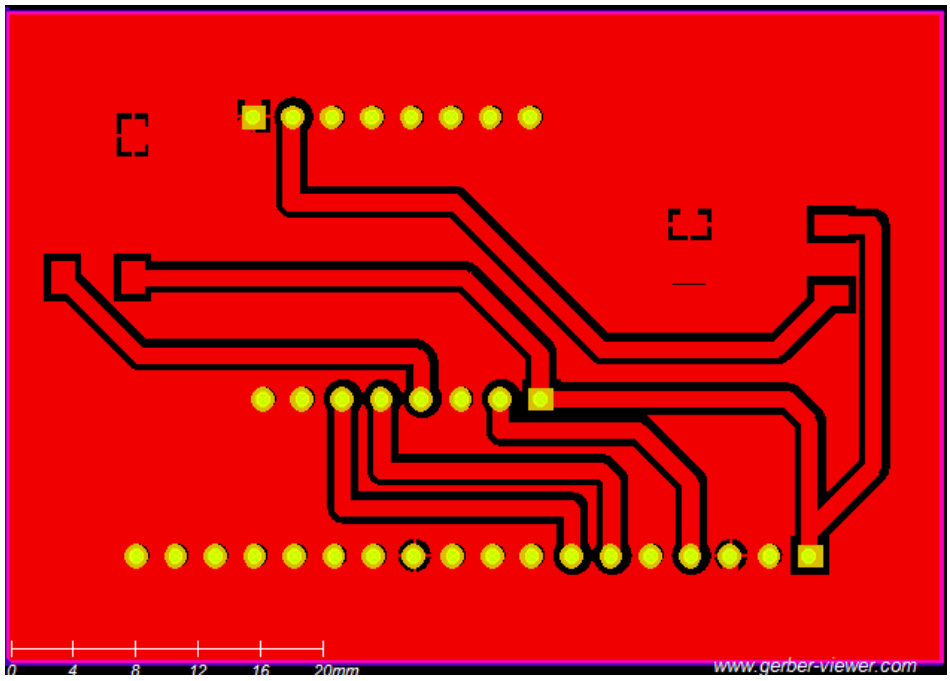

Image 13 : PCB de la carte

La liaison d'une patte des boutons avec la pin 5v était une erreur , afin d'éviter d'imprimer une autre carte je n'ai pas soudé ces pattes .Cela estfonctionnel , mais l'alimentation du circuit FONA 3G et l'afficheu MicroViewr ne reste pas maintenue sauf si je garde les circuits dans une position précise même après les avoir soudés . Ceci peut être dû aux broches des header non utilisées et par conséquent non soudées qui poussent la carte à ne pas bien tenir les composants .

Résultat :

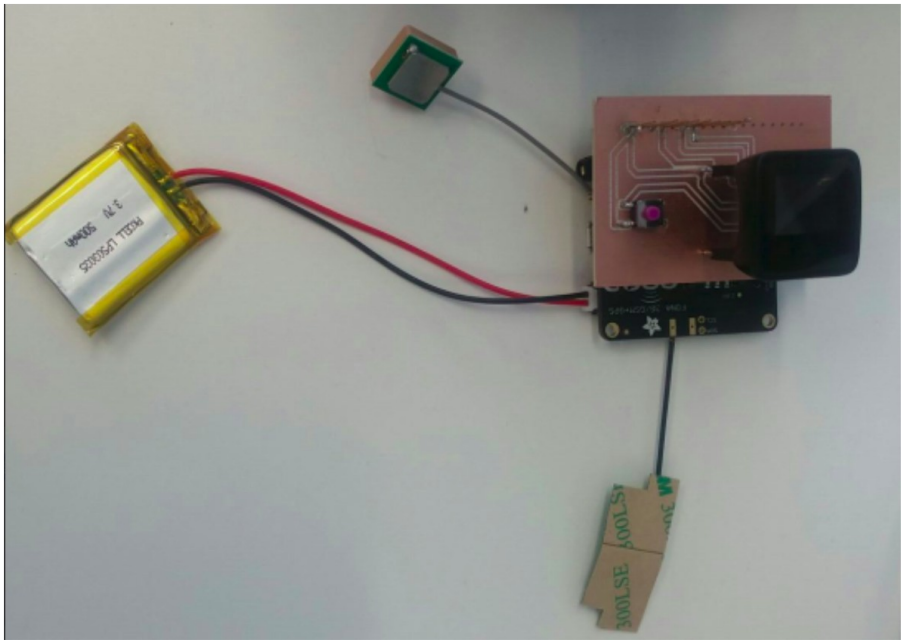

Image 14: Carte réalisé & montage final

Conclusion

J'ai choisi ce projet qui regroupe la programmation à objet et la télécommunication afin de mettre en pratique les acquis dans ces disciplines . Il m'a permi d'aller de la théorie en télécommunication à la réalisation des services GSM ainsi que la géolocalisation GPS en passant par la programmation des modules correspondants.

La partie où j'ai rencontré des difficultées est la géolocalisation . Cela m'a amenée à découvrir le principe de fonctionnement des différents moyens d'obtention d'une géolocalisation tel que : la géolocalisation GSM par Cell-ID , TDOA ou triangulation et la géolocalisation par Internet qui consiste à localiser une adresse IP . L'application de ces méthodes m'a poussée à modifier le code source fourni par le constructeur à l'encontre de son utilisation directe .

L'expérience de ces moyens de géolocalisation n'a pas abouti à son but, ce qui m'a conduit à exploiter la partie matérielle . Bien que la solution proposée pour cet obstacle n'était pas adéquate au projet , j'en ai tiré la distinction des types d'antennes GPS .

L'objectif d'avoir un montage semblable le plus réellement possible à une montre , impose une utilisation minimale de matériels et de petites dimensions . Cela n'était pas serviable afin d'améliorer mes compétences en conception d'une carte électronique . Cette tâche n'était pas aussi importante que celles précédentes pour le projet , pourtant elle l'était pour mon apprentissage. Malgré le temps passé pour une carte simple, il a fallu assimiler toutes les configurations valides pour obtenir une carte réalisable.

Finalement la valeur ajoutée de ce projet était plus pédagogique . J'ai appris à ne pas se fier des grandes lignes lors d'un choix , de se concentrer en premier lieu sur le minimum à réaliser avant de penser à l'amélioration du projet puis avancer étape par étape pour des résultats meilleurs .

Annexes:

Semaine 1

```
#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(3, 2);
Adafruit_GPS GPS(&mySerial);
String NMEA1;
String NMEA2;
char c;

void setup() {
Serial.begin(115200);
GPS.begin(9600);
GPS.sendCommand("$PGCMD,33,0*6D");
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);
delay(1000); }

void loop(){
readGPS();
}

void readGPS(){
clearGPS();

while(!GPS.newNMEAreceived()) {
c=GPS.read();
}
GPS.parse(GPS.lastNMEA());
NMEA1=GPS.lastNMEA();
while(!GPS.newNMEAreceived()) {
c=GPS.read();
}

GPS.parse(GPS.lastNMEA());
NMEA2=GPS.lastNMEA();

Serial.println(NMEA1);
Serial.println(NMEA2);
Serial.println("");
Serial.println(NMEA1);
```

```
Serial.println(NMEA2);
Serial.println("");
Serial.print("\nTime: ");
Serial.print(GPS.hour, DEC); Serial.print(':');
Serial.print(GPS.minute, DEC); Serial.print(':');
Serial.print(GPS.seconds, DEC); Serial.print('.');
Serial.println(GPS.milliseconds);
Serial.print("Date: ");
Serial.print(GPS.day, DEC); Serial.print('/');
Serial.print(GPS.month, DEC); Serial.print("/20");
Serial.println(GPS.year, DEC);
Serial.print("Fix: "); Serial.print((int)GPS.fix);
Serial.print(" quality: "); Serial.println((int)GPS.fixquality);

if (GPS.fix) {
Serial.print("Location: ");
Serial.print(GPS.latitude, 4); Serial.print(GPS.lat);
Serial.print(", ");
Serial.print(GPS.longitude, 4); Serial.println(GPS.lon);
Serial.print("Location (in degrees, works with Google Maps): ");
Serial.print(GPS.latitudeDegrees, 4);
Serial.print(", ");
Serial.println(GPS.longitudeDegrees, 4);
Serial.print("Speed (knots): "); Serial.println(GPS.speed);
Serial.print("Angle: "); Serial.println(GPS.angle);
Serial.print("Altitude: "); Serial.println(GPS.altitude);
Serial.print("Satellites: "); Serial.println((int)GPS.satellites);
}
}

void clearGPS() {

while(!GPS.newNMEAreceived())
{
c=GPS.read();
}

GPS.parse(GPS.lastNMEA());

while(!GPS.newNMEAreceived()) {

c=GPS.read();
}

GPS.parse(GPS.lastNMEA());}
```

Semaine 2

```
//Ce code permet d'envoyer la position par SMS une fois le bouton est appuyé
#include <GSM.h>
#include <Adafruit_GPS.h>
```

```
#include<AltSoftSerial.h>//utilisation de AltSoftSerial pour eviter le conflit des librairies GSM et GPS pour utiliser SoftwareSerial.h
AltSoftSerial mySerial(3, 2);
Adafruit_GPS GPS(&mySerial);// objet GPS

#define PINNUMBER "1234"// code pin
GSM gsmAccess;
GSM_SMS sms;// objet sms
const int buttonPin = 8;
int buttonState = 0;

void setup() {
pinMode(buttonPin, INPUT_PULLUP);
Serial.begin(9600);
boolean notConnected = true;
Serial.println("Initialisation GSM en cours ...");

//test d'initialisation GSM

while (notConnected) {
if (gsmAccess.begin(PINNUMBER) == GSM_READY) {
notConnected = false;
}
else {
Serial.println("non connecté");
delay(1000);
}
Serial.println("GSM initialisé: appuyez sur le bouton pour envoyé la position par sms ");
}

}
void loop() {

buttonState = digitalRead(buttonPin);
char* remoteNum="0753896813"; // num destinataire
String latitude = String(GPS.latitudeDegrees);
String longitude= String(GPS.longitudeDegrees);
// envoie du message

if (buttonState == LOW) {
Serial.println("envoie de la position en cours ...\n");
sms.beginSMS(remoteNum);
sms.print("lien google maps: http://maps.google.com/?q="+latitude+","+longitude);
sms.endSMS();
Serial.println("\nPosition envoyée:lien google maps: http://maps.google.com/?q="+latitude+","+longitude);
}
}
```

Semaine 4_1

```
// Affichage des flèches sur LCD
#include <Adafruit_GFX.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_ILI9341.h>
#include <Adafruit_STMPE610.h>

#define STMPE_CS 8
Adafruit_STMPE610 ts = Adafruit_STMPE610(STMPE_CS);
#define TFT_CS 10
#define TFT_DC 9
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

#define D_X0 160
#define D_Y0 260
#define D_X1 60
#define D_Y1 260
#define D_X2 110
#define D_Y2 300


#define G_X0 160
#define G_Y0 60
#define G_X1 60
#define G_Y1 60
#define G_X2 110
#define G_Y2 20


#define H_X0 60
#define H_Y0 100
#define H_X1 60
#define H_Y1 220
#define H_X2 20
#define H_Y2 160


#define B_X0 180
#define B_Y0 100
#define B_X1 180
#define B_Y1 220
#define B_X2 220
#define B_Y2 160

void droite()
{
tft.fillTriangle(D_X0,D_Y0,D_X1,D_Y1,D_X2,D_Y2,ILI9341_GREEN);
tft.setCursor(D_X1+13,D_Y0);
```

```
tft.setTextColor(ILI9341_BLACK);
tft.setTextSize(2);
tft.println("droite");
}

void gauche()
{
tft.fillTriangle(G_X0,G_Y0,G_X1,G_Y1,G_X2,G_Y2,ILI9341_GREEN);
tft.setCursor(G_X1+13,G_Y1-13);
tft.setTextColor(ILI9341_BLACK);
tft.setTextSize(2);
tft.println("gauche");
}
void haut()
{
tft.fillTriangle(H_X0,H_Y0,H_X1,H_Y1,H_X2,H_Y2,ILI9341_GREEN);
}
void bas()
{

tft.fillTriangle(B_X0,B_Y0,B_X1,B_Y1,B_X2,B_Y2,ILI9341_GREEN);
}

void setup(void)
{
Serial.begin(9600);
tft.begin();
if (!ts.begin()) {
Serial.println("Unable to start touchscreen.");
}
else {
Serial.println("Touchscreen started.");
}

tft.fillScreen(ILI9341_PINK);
}

void loop()
{
droite();
gauche();
haut();
bas();
}
```

Semaine 5

```
//Code pour faire clignoter la bonne fleche selon l'orientation
#include <Adafruit_GPS.h>
#include <AltSoftSerial.h>
```

```
#include <Adafruit_GFX.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_ILI9341.h>
#include <Adafruit_STMPE610.h>


#define STMPE_CS 8
Adafruit_STMPE610 ts = Adafruit_STMPE610(STMPE_CS);
#define TFT_CS 10
#define TFT_DC 9
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);


#define D_X0 160
#define D_Y0 260
#define D_X1 60
#define D_Y1 260
#define D_X2 110
#define D_Y2 300

#define G_X0 160
#define G_Y0 60
#define G_X1 60
#define G_Y1 60
#define G_X2 110
#define G_Y2 20

#define H_X0 60
#define H_Y0 100
#define H_X1 60
#define H_Y1 220
#define H_X2 20
#define H_Y2 160


#define B_X0 180
#define B_Y0 100
#define B_X1 180
#define B_Y1 220
#define B_X2 220
#define B_Y2 160

#define lat_domicile 50.61108650000001
#define longi_domicile 3.1490148000000318

AltSoftSerial mySerial(3, 2);
Adafruit_GPS GPS(&mySerial);
String NMEA1;
String NMEA2;
```

```
char c;
void setup()
{
// Serial.begin(9600);
tft.begin();
if (!ts.begin()) {
Serial.println("Unable to start touchscreen.");
}
else {
Serial.println("Touchscreen started.");
}

tft.fillScreen(ILI9341_PINK);
Serial.begin(115200);
GPS.begin(9600);
GPS.sendCommand("$PGCMD,33,0*6D");
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);
delay(1000);
}
void loop() {

Serial.print("lien google maps: http://maps.google.com/?q=");
Serial.print(GPS.latitudeDegrees, 4);
Serial.print(", ");
Serial.println(GPS.longitudeDegrees, 4);

if (lat_domicile-GPS.latitudeDegrees>0){
haut();
}
else{
bas();
}
if(longi_domicile-GPS.longitudeDegrees>0){
droite();
}
else{
gauche();
}
}


void droite()
{
tft.fillTriangle(D_X0,D_Y0,D_X1,D_Y1,D_X2,D_Y2,ILI9341_GREEN);
tft.setCursor(D_X1+13,D_Y0);
tft.setTextColor(ILI9341_BLACK);
tft.setTextSize(2);
tft.println("droite");
//delay(5000);
```

```
}

void gauche()
{
tft.fillTriangle(G_X0,G_Y0,G_X1,G_Y1,G_X2,G_Y2,ILI9341_GREEN);
tft.setCursor(G_X1+13,G_Y1-13);
tft.setTextColor(ILI9341_BLACK);
tft.setTextSize(2);
tft.println("gauche");
}
void haut()
{
tft.fillTriangle(H_X0,H_Y0,H_X1,H_Y1,H_X2,H_Y2,ILI9341_GREEN);
}
void bas()
{

tft.fillTriangle(B_X0,B_Y0,B_X1,B_Y1,B_X2,B_Y2,ILI9341_GREEN);
}
```

Semaine 6_1

```
// Affichage des flèches sur MicroViewer
#include <MicroView.h>


void setup() {
Serial.begin(9600);
uView.begin();
uView.clear(PAGE);
}

void loop() {
uView.clear(PAGE);

haut();

delay(1000);
uView.clear(PAGE);

uView.clear(PAGE);


droite();
delay(1000);
uView.clear(PAGE);

uView.clear(PAGE);
```

```
delay(1000);
uView.clear(PAGE);
bas();
delay(1000);
uView.clear(PAGE);
gauche();
delay(1000);
uView.clear(PAGE);

}
void droite(){
uView.rectFill(10,18,30,12);
uView.lineV(40,5,37);
uView.line(40,5,55,24);
uView.line(55,24,40,43);
uView.display();
}
void gauche(){
uView.rectFill(25,18,30,12);
uView.lineV(25,5,37);
uView.line(25,5,10,24);
uView.line(10,24,25,43);
uView.display();
}
void haut(){
uView.rectFill(25,15,12,30);
uView.lineH(15,15,32);
uView.line(15,15,31,1);
uView.line(31,1,47,15);
uView.display();
}
void bas(){
uView.rectFill(25,1,12,30);
uView.lineH(15,31,32);
uView.line(15,31,31,46);
uView.line(31,46,47,31);
uView.display();
}
```

Semaine 6₂

```
// Affichage de la bonne flèche d'orientation
#include <AltSoftSerial.h>
#include <SPI.h>
#include <Wire.h>
#include <MicroView.h>
#include <Adafruit_GPS.h>


#define lat_domicile 50.61108650000001
#define longi_domicile 3.1490148000000318
```

```
AltSoftSerial mySerial(3, 2);
Adafruit_GPS GPS(&mySerial);
String NMEA1;
String NMEA2;
char c;
void setup()
{
// Serial.begin(9600);
uView.begin();
uView.clear(PAGE);
Serial.begin(115200);
GPS.begin(9600);
GPS.sendCommand("$PGCMD,33,0*6D");
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);
delay(1000);
}
void loop() {

Serial.print("lien google maps: http://maps.google.com/?q=");
Serial.print(GPS.latitudeDegrees, 4);
Serial.print(", ");
Serial.println(GPS.longitudeDegrees, 4);
while(lat_domicile-GPS.latitudeDegrees!=0){
if (lat_domicile-GPS.latitudeDegrees>0){
haut();
uView.clear(PAGE);
}
else{
bas();
uView.clear(PAGE);
}
}
while(longi_domicile-GPS.longitudeDegrees!=0){
if(longi_domicile-GPS.longitudeDegrees>0){
droite();
uView.clear(PAGE);
}
else{
gauche();
uView.clear(PAGE);
}
}
}

void droite(){
uView.rect(10,18,30,12);
uView.lineV(40,5,37);
uView.line(40,5,55,24);
uView.line(55,24,40,43);
```

```
uView.display();
}
void gauche(){
uView.rect(25,18,30,12);
uView.lineV(25,5,37);
uView.line(25,5,10,24);
uView.line(10,24,25,43);
uView.display();
}
void haut(){
uView.rect(25,15,12,30);
uView.lineH(15,15,32);
uView.line(15,15,31,1);
uView.line(31,1,47,15);
uView.display();
}
void bas(){
uView.rect(25,1,12,30);
uView.lineH(15,31,32);
uView.line(15,31,31,46);
uView.line(31,46,47,31);
uView.display();
}
```

Semaine 9

```
//Géolocalisation par FONA

#include "Adafruit_FONA.h"


#define FONA_RX 3
#define FONA_TX 2
#define FONA_RST 4


#include <SoftwareSerial.h>
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

Adafruit_FONA_3G fona = Adafruit_FONA_3G(FONA_RST);


void setup() {

while (! Serial);

Serial.begin(115200);
Serial.println(F("Initialisation FONA."));
```

```
fonaSerial->begin(4800);
if (! fona.begin(*fonaSerial)) {
Serial.println(F("FONA introuvable"));
while(1);
}
Serial.println(F("FONA is OK"));
Serial.println(F("Enabling GPS..."));
fona.enableGPS(true);
}

void loop() {
delay(2000);

float latitude, longitude, speed_kph, heading, speed_mph, altitude;

// if you ask for an altitude reading, getGPS will return false if there isn't a 3D fix
boolean gps_success = fona.getGPS(&latitude, &longitude, &speed_kph, &heading,
&altitude);

if (gps_success) {

Serial.print("GPS lat:");
Serial.println(latitude, 6);
Serial.print("GPS long:");
Serial.println(longitude, 6);
Serial.print("GPS altitude:");
Serial.println(altitude);

} else {
Serial.println("EN ATTENTE DU FIX 3D...");
}
}
```

Semaine 10_11

```
//Géolocalisation + SMS + Appel par FONA
#include "Adafruit_FONA.h"
#include <SoftwareSerial.h>
#include <stdio.h>
#include <string.h>
#include <MicroView.h>

#define FONA_RX 3
#define FONA_TX 2
#define FONA_RST 4
#define lat_domicile 50.61108650000001
#define longi_domicile 3.1490148000000318

SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
```

```
SoftwareSerial *fonaSerial = &fonaSS;

Adafruit_FONA_3G fona = Adafruit_FONA_3G(FONA_RST);
float latitude, longitude, speed_kph, heading, speed_mph, altitude;

boolean gps_success;
uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);

const int Bouton1=5;
int EtatBouton1;
const int Bouton2=A0;
int EtatBouton2;

void setup() {
uView.begin();
uView.clear(PAGE);
pinMode(Bouton1,INPUT_PULLUP);
pinMode(Bouton2,INPUT_PULLUP);
Serial.begin(115200);
Serial.println(F("FONA set baudrate"));

Serial.println(F("First trying 115200 baud"));
// start at 115200 baud
fonaSerial->begin(115200);
fona.begin(*fonaSerial);
// send the command to reset the baud rate to 4800
fona.setBaudrate(4800);
// restart with 4800 baud
fonaSerial->begin(4800);
Serial.println(F("Initializing @ 4800 baud..."));
if (! fona.begin(*fonaSerial)) {
Serial.println(F("Couldn't find FONA"));
while(1);
}
Serial.println(F("FONA is OK"));
}

void loop() {
EtatBouton1=digitalRead(Bouton1);
EtatBouton2=digitalRead(Bouton2);
if(EtatBouton1==LOW ){
WriteSMS();
delay(5000);
makingCall();
}


}
void WriteSMS () {
float latitude;
```

```
float longitude;
gps_success = fona.getGPS(&latitude, &longitude, &speed_kph, &heading, &altitude);
char message[80];
char* num="0753896813";
char Slatitude[20];
char Slongitude[20];
dtostrf(latitude,8,5,Slatitude);
dtostrf(longitude,8,5,Slongitude);
//if(gps_success){
strcpy(message,"lien google maps: http://maps.google.com/?q=");
strcat(message,Slatitude);
strcat(message,",");
strcat(message,Slongitude);
puts(message);
Serial.println(message);
fona.sendSMS(num,message);
delay(5000);
/*}
else{
Serial.println("Waiting for FIX..... ");
}*/
}


void makingCall(){
char num[20]="0753896813";
fona.callPhone(num);
}
```

Semaine 12

```
// Géolocalisation seul pour tentative d'avoir un FIX bon
#include "Adafruit_FONA.h"
#include <SoftwareSerial.h>
#include <stdio.h>
#include <string.h>


#define FONA_RX 3
#define FONA_TX 2
#define FONA_RST 4

SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

Adafruit_FONA_3G fona = Adafruit_FONA_3G(FONA_RST);
float latitude, longitude, speed_kph, heading, speed_mph, altitude;

boolean gps_success;
```

```
uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);

const int Bouton1=5;
int EtatBouton1;
const int Bouton2=A0;
int EtatBouton2;

void setup() {
pinMode(Bouton1,INPUT_PULLUP);
pinMode(Bouton2,INPUT_PULLUP);
Serial.begin(115200);
Serial.println(F("FONA set baudrate"));

Serial.println(F("First trying 115200 baud"));
// start at 115200 baud
fonaSerial->begin(115200);
fona.begin(*fonaSerial);
// send the command to reset the baud rate to 4800
fona.setBaudrate(4800);
// restart with 4800 baud
fonaSerial->begin(4800);
Serial.println(F("Initializing @ 4800 baud..."));
if (! fona.begin(*fonaSerial)) {
Serial.println(F("Couldn't find FONA"));
while(1);
}
Serial.println(F("FONA is OK"));
}

void loop() {
EtatBouton1=digitalRead(Bouton1);
EtatBouton2=digitalRead(Bouton2);
if(EtatBouton1==LOW || EtatBouton2==LOW ){
Serial.println(EtatBouton1);
Serial.println(EtatBouton2);
WriteSMS();
}
}
void WriteSMS () {
float latitude;
float longitude;
gps_success = fona.getGPS(&latitude, &longitude, &speed_kph, &heading,
&altitude);
char message[80];
char* num="0753896813";
char Slatitude[20];
char Slongitude[20];
dtostrf(latitude,8,5,Slatitude);
dtostrf(longitude,8,5,Slongitude);
if(gps_success){
```

```
strcpy(message,"lien google maps: http://maps.google.com/?q=");
strcat(message,Slatitude);
strcat(message,",");
strcat(message,Slongitude);
puts(message);
Serial.println(message);
fona.sendSMS(num,message);
delay(5000);
}
else{
Serial.println("Impossible d'envoyer SMS , en attente du FIX ..... ");
}
}
```

Code global

```
// Code global : Géolocalisation + SMS + Appel + Orientation + Affichage de l'heure
#include "Adafruit_FONA.h"
#include <SoftwareSerial.h>
#include <stdio.h>
#include <string.h>
#include <MicroView.h>
#include <Time.h>

#define FONA_RX 3
#define FONA_TX 2
#define FONA_RST 4
#define lat_domicile 50.61108650000001
#define longi_domicile 3.1490148000000318
#define CLOCK_SIZE 23
#define HOUR 01
#define MINUTE 01
#define SECOND 00
#define DAY 11
#define MONTH 5
#define YEAR 2018

SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

Adafruit_FONA_3G fona = Adafruit_FONA_3G(FONA_RST);
float latitude, longitude, speed_kph, heading, speed_mph, altitude;
boolean gps_success;
const int Bouton1=5;
int EtatBouton1;
const int Bouton2=A0;
int EtatBouton2;
const uint8_t maxW = uView.getLCDWidth();
const uint8_t midW = maxW/2;
const uint8_t maxH = uView.getLCDHeight();
```

```
const uint8_t midH = maxH/2;

void setup() {
setTime(HOUR, MINUTE, SECOND, DAY, MONTH, YEAR);
uView.begin();
uView.clear(PAGE);
uView.display();

drawFace();
pinMode(Bouton1,INPUT_PULLUP);
pinMode(Bouton2,INPUT_PULLUP);
Serial.begin(115200);
Serial.println(F("FONA set baudrate"));

Serial.println(F("First trying 115200 baud"));
// start at 115200 baud
fonaSerial->begin(115200);
fona.begin(*fonaSerial);
// send the command to reset the baud rate to 4800
fona.setBaudrate(4800);
// restart with 4800 baud
fonaSerial->begin(4800);
Serial.println(F("Initializing @ 4800 baud..."));
if (! fona.begin(*fonaSerial)) {
Serial.println(F("Couldn't find FONA"));
while(1);
}
Serial.println(F("FONA is OK"));
fona.enableGPS(true);
}

void loop() {
drawTime();
EtatBouton1=digitalRead(Bouton1);
EtatBouton2=digitalRead(Bouton2);
if(EtatBouton1==LOW ){
Serial.println(EtatBouton1);
WriteSMS();
delay(5000);
makingCall();
}
if(EtatBouton2==LOW && gps_success){
Orientation();
}
}
void WriteSMS () {
float latitude;
float longitude;
gps_success = fona.getGPS(&latitude, &longitude, &speed_kph, &heading,
&altitude);
```

```
char message[80];
char* num="0753896813";
char Slatitude[20];
char Slongitude[20];
dtostrf(latitude,8,5,Slatitude);
dtostrf(longitude,8,5,Slongitude);
//if(gps_success){ // A dé-commenter si on veut que l'envoi du SMS soit sous condition d'obtention des coocrdonnée GPS
strcpy(message,"lien google maps: http://maps.google.com/?q=");
strcat(message,Slatitude);
strcat(message,",");
strcat(message,Slongitude);
puts(message);
Serial.println(message);
fona.sendSMS(num,message);
delay(5000);
/*}
else{
Serial.println("Waiting for FIX..... ");
}*/
}


void GetGPS() {
//gps_success = 0;
/*fonaSerial->begin(4800);
if (! fona.begin(*fonaSerial)) {
while (1);
}
fona.enableGPS(true);*/
//while(!gps_success){
delay(2000);
gps_success = fona.getGPS(&latitude, &longitude, &speed_kph, &heading, &altitude);
if (gps_success) {
Serial.print("GPS lat:");
Serial.println(latitude, 6);
Serial.print("GPS long:");
Serial.println(longitude, 6);
} else{
Serial.println("En attente du FIX");
}
}
void makingCall(){
char num[20]="0753896813";
//sous condition de bouton//
fona.callPhone(num);
}
void Orientation(){
while(lat_domicile-latitude!=0){
```

```
if (lat_domicile-latitude>0){
haut();
uView.clear(PAGE);
}
else{
bas();
uView.clear(PAGE);
}
}
while(longi_domicile-longitude!=0){
if(longi_domicile-longitude>0){
droite();
uView.clear(PAGE);
}
else{
gauche();
uView.clear(PAGE);
}
}
if(lat_domicile-latitude==0 && longi_domicile-longitude==0){
//afficher sur µview HOME
}
}



void droite(){
uView.rectFill(10,18,30,12);
uView.lineV(40,5,37);
uView.line(40,5,55,24);
uView.line(55,24,40,43);
uView.display();
}
void gauche(){
uView.rectFill(25,18,30,12);
uView.lineV(25,5,37);
uView.line(25,5,10,24);
uView.line(10,24,25,43);
uView.display();
}
void haut(){
uView.rectFill(25,15,12,30);
uView.lineH(15,15,32);
uView.line(15,15,31,1);
uView.line(31,1,47,15);
uView.display();
}
void bas(){
uView.rectFill(25,1,12,30);
uView.lineH(15,31,32);
```

```
uView.line(15,31,31,46);
uView.line(31,46,47,31);
uView.display();
}
void drawTime()
{
static boolean firstDraw = false;
static unsigned long mSec = millis() + 1000;
static float degresshour, degressmin, degresssec,
hourx, houry, minx, miny, secx, secy;
// If mSec
if (mSec != (unsigned long)second())
{
// First time draw requires extra line to set up XOR's:
if (firstDraw)
{
uView.line(midW, midH, 32 + hourx, 24 + houry, WHITE, XOR);
uView.line(midW, midH, 32 + minx, 24 + miny, WHITE, XOR);
uView.line(midW, midH, 32 + secx, 24 + secy, WHITE, XOR);
}
// Calculate hour hand degrees:
degresshour = (((hour() * 360) / 12) + 270) * (PI / 180);
// Calculate minute hand degrees:
degressmin = (((minute() * 360) / 60) + 270) * (PI / 180);
// Calculate second hand degrees:
degresssec = (((second() * 360) / 60) + 270) * (PI / 180);

// Calculate x,y coordinates of hour hand:
hourx = cos(degresshour) * (CLOCK_SIZE / 2.5);
houry = sin(degresshour) * (CLOCK_SIZE / 2.5);
// Calculate x,y coordinates of minute hand:
minx = cos(degressmin) * (CLOCK_SIZE / 1.4);
miny = sin(degressmin) * (CLOCK_SIZE / 1.4);
// Calculate x,y coordinates of second hand:
secx = cos(degresssec) * (CLOCK_SIZE / 1.1);
secy = sin(degresssec) * (CLOCK_SIZE / 1.1);

// Draw hands with the line function:
uView.line(midW, midH, midW+hourx, midH+houry, WHITE, XOR);
uView.line(midW, midH, midW+minx, midH+miny, WHITE, XOR);
uView.line(midW, midH, midW+secx, midH+secy, WHITE, XOR);
firstDraw = true;
uView.display();
}
}


void drawFace()
{
uView.setFontType(0);
```

```
uint8_t fontW = uView.getFontWidth();
uint8_t fontH = uView.getFontHeight();
uView.setCursor(midW-fontW-1, midH-CLOCK_SIZE+1);
uView.print(12); // Print the "12"
uView.setCursor(midW-(fontW/2)-1, midH+CLOCK_SIZE-fontH-1);
uView.print(6); // Print the "6"
uView.setCursor(midW-CLOCK_SIZE+1, midH-fontH/2);
uView.print(9); // Print the "9"
uView.setCursor(midW+CLOCK_SIZE-fontW-2, midH-fontH/2);
uView.print(3); // Print the "3"
uView.circle(midW-1, midH-1, CLOCK_SIZE);
uView.display();
}
```

Bibliographie :

datasheet FONA 3G

Commandes AT du SIMCOM5320

hardware design SIMCOM5320