GEII
Département Génie Électrique
& Informatique Industrielle
IUT Belfort-Montbéliard

Transmission des Données CAN par GSM

RAPPORT DE PROJET

GILBERT Lucas
BRIBI Riad

Licence professionnel VEGA

GUSTIN Fréderic

UNIVERSITÉ DE FRANCHE-COMTÉ

IUT Belfort-Montbéliard

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD
Tél. +33 (0)3 84 58 77 00 • iut-bm.univ-fcomte.fr

Table des matières

Remerciements :

Nos remerciements vont à notre encadrant technique Mr GUSTIN pour son aide au lancement du projet et qui nous a aidé durant le déroulement de celui-ci. Ses conseils nous ont permis de nous orienter dans la bonne direction, ce qui a était extrêmement instructif.

Nous remercions également toutes l'équipe pédagogique du département GEII de l'IUT Belfort-Montbéliard et les intervenants professionnels responsables de la formation licence professionnelle VEGA : Véhicules Électroniques et Gestion des Automatismes pour avoir assuré la partie théorique de celle-ci.

Introduction du Projet :

Notre objectif est la réalisation d'un système de récolte d'informations à distance d'un système CAN (automobile).

Fonction à réaliser :

Etre dans la capacité d'envoyer des données via une puce GSM sur un téléphone utilisateur ou une autre puce GSM.

Ces données sont des trames CAN extraite du flux, traité et affiché pour l'utilisateur avec le numéro d'identifiant de là cette trame et sont contenue.

De cette manière les informations souhaitées seront traçable à distance sur un système CAN sans avoir à brancher et débranché un système complet pour faire une vérification spécification.

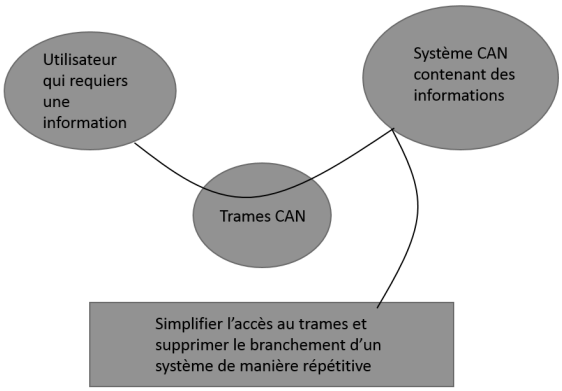
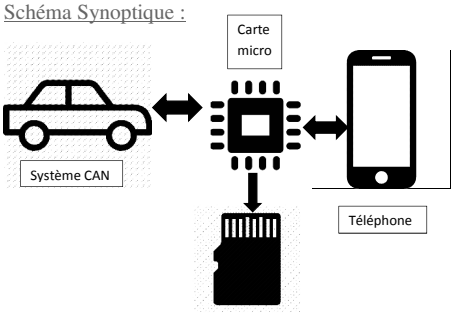


Schéma Synoptique :

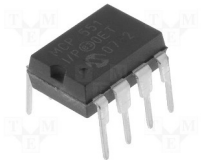


Ce Schéma synoptique permet de rendre compte du fonctionnement du système et des différents liens entre chaque module de celui-ci.

Ici la carte est liée au système CAN de manière physique et est donc embarquée avec celui-ci, l'intégralité des données sélectionné sont stocké sur la mémoire SD ajouté à la carte, le téléphone lui reçoit à intervalle régulière des messages avec le contenue des trames et leur identifiant en fonction des réglages du programme.

Les trois composants principaux de ce système sont :

-MCP2551 (convertisseur CAN)



-SIM800L (Puce GSM)

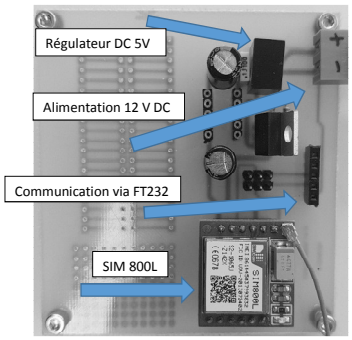
18F4580 (microcontrôleur)

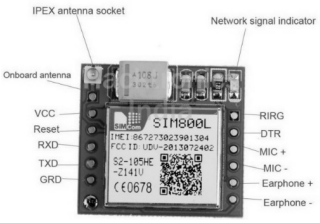
Réalisation du Projet :

Premier pas avec le module :

Dans un premier temps nous nous sommes concentré sur la mise en œuvre du module SIM 800L

Avec une carte donnée par notre Professeur tuteur Mr GUSTIN, celle-ci nous a permis avec Tiny Bootloader (un petit logiciel applicatif) et un module FT232 de communiquer sur la liaison série du module SIM 800L, avec cette solution nous avons été capable de mettre en place le protocole qui permet l'envoie d'un message avec ce module à travers les commandes AT.





Le SIM800L est alimenté entre son VCC et sa masse. Il émet ses données via sa patte « TXD : transmission de donnée ». Il reçoit les données via sa patte « RXD : réception de données ». Pour son émission de donnée il possède un connecteur pour brancher une antenne déportée.

La partie de droite avec le micro et le haut-parleur n'a pas était utilisé lors de notre projet.

En dessous du SIM800L on peut insérer une puce GSM

Commandes AT :

Ce protocole utilise une solution qui s'appelle les commandes AT ces commandes communes à tous les systèmes GSM du monde permet donc d'envoyer des ordres et de recevoir des informations sur l'état du module.

Ces commandes AT dans un premier temps envoyer à travers le logiciel TinyBLD qui nous permettaient d'envoyer des requêtes une par une. Cela nous a donc permis de valider le fonctionnement du SIM800L.

« Exemple de commande AT » :

```
34
35      printf("AT+CPIN=1234");
36      printf("%c",b);
37      delay_ms(500);
38      printf("AT+COPS?");
39      printf("%c",b);
40      delay_ms(500);
41      printf("AT+CNUM");
42      printf("%c",b);
43      delay_ms(500);
44      printf("AT+CMGF=1");
45      printf("%c",b);
46      delay_ms(500);
47      printf("AT+CMGs=");
48      printf("%c",c);
49      printf("+33770284936");
50      printf("%c",c);
51      printf("%c",b);
52      delay_ms(500);
53      printf("Identifiant: ");
54      printf("%3x",rx_id);
55      printf("%c",b);
56      printf("Data: ");
57      printf("%x",buffer[0]);
58      printf("%x",buffer[1]);
59      printf("%x",buffer[2]);
60      printf("%x",buffer[3]);
61      printf("%x",buffer[4]);
62      printf("%x",buffer[5]);
63      printf("%x",buffer[6]);
64      printf("%x",buffer[7]);
65      delay_ms(500);
66      printf("%c",a);
67      delay_ms(500);
68      printf(0x1A);
69      delay_ms(500);
70       }
```

Entrée du code PIN

Permet de verifier si il y a bien un réseau

Permet de récupérer son numéro de téléphone

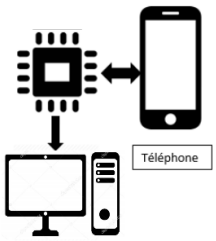
Entrée en mode message

Passage en mode Sms et saisie du numéro du destinataire.

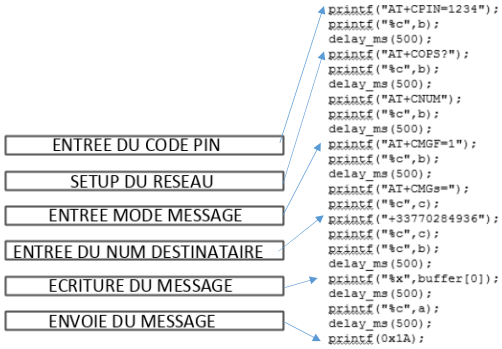
Entrée du message à envoyé

Envoie du SMS avec l'envoie de 1A en héxa

Les premiers tests de validation de commande AT ont été réalisé dans le contexte si contre, nous utilisions le PC comme donneur de commande, qui pilote la SIM800L avec pour objectif d'envoyer un message.



Exemple de commande AT :



Premier test en C :

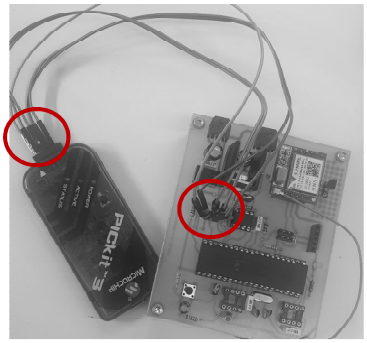
L'étape suivante était donc de passer à un système automatique qui envoie les informations en fonction de temporisation précise et non plus sur l'impulsion de l'utilisateur.

Nous nous sommes donc tourné vers la solution proposée par notre professeur une carte composé d'un microcontrôleur ici un 18f4580 celui utilisé lors de notre formation en C.

Nous sommes donc avec un matériel connu mais une carte qui n'est pas de notre réalisation.

L'objectif à ce moment précis était d'envoyer sur certaines sorties du microcontrôleur la suite de commandes AT de notre protocole.

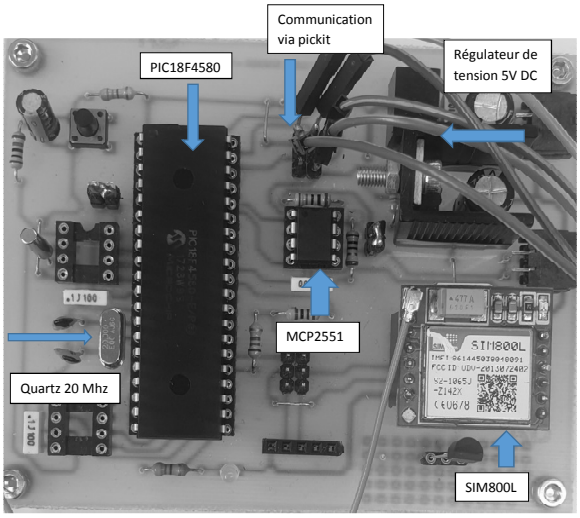
Cette étape nous a porté préjudice, le groupe précédant qui avais réalisé la carte n'avais pas fait une liaison correcte pour le PICKit-3 ce qui nous a poser soucis pendant quelques heures, il nous était pas possible d'avancer sur le programme car le test de celui-ci était impossible.



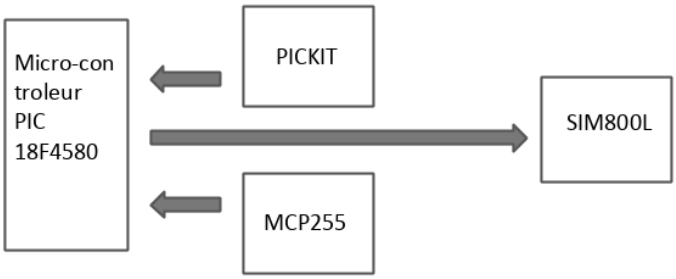
Une fois ce problème réglé nous avons été en mesure de valider l'envoie de chacune des commandes sur les sorties escomptées.

Ajout Microcontrôleur et SIM800L :

La suite logique des choses était de mettre en fonction le microcontrôleur et le SIM800L pour automatisé l'envoie du message.



Cette partie nous a également apporté son lot de soucis pour la suite, en réalité le code C ne fonctionnais et ne nous donnais les informations désirées que lorsque la fréquence de référence du programme était donnée à 1Mhz contre 20Mhz avec le quartz utilisé sur la carte.



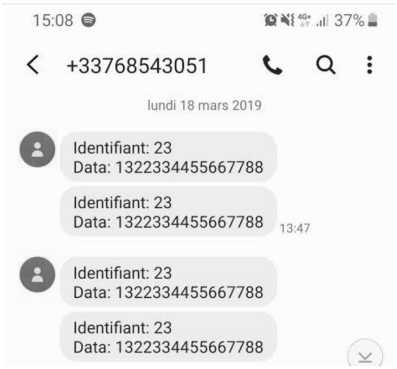
Voilà le schéma synoptique qui explique l'objectif de cette assemblage et la réalisation des tests.

(Les flèches indique le sens du flux de données que nous cherchons à mettre en place et pas le flux absolu de communication entre les organes).

L'objectif principal était de mettre en place une liaison entre le PIC 18f4580 et le SIM800L, cette liaison a donc été testé dans un premier temps de manière 'fictive', utilisation d'un module FT232 pour écouter en lieu et place du SIM800L pour vérifier la présence et la conformité des informations envoyer par le microcontrôleur.

Une fois ces informations validé et vérifié nous avons été en mesure d'envoyer des messages programmer sur le microcontrôleur et de les recevoir sur le téléphone mobile

L'automatisation s'est bien passée et nous étions en mesure d'envoyer un message de manière totalement automatisé à travers une variable et des commandes AT.



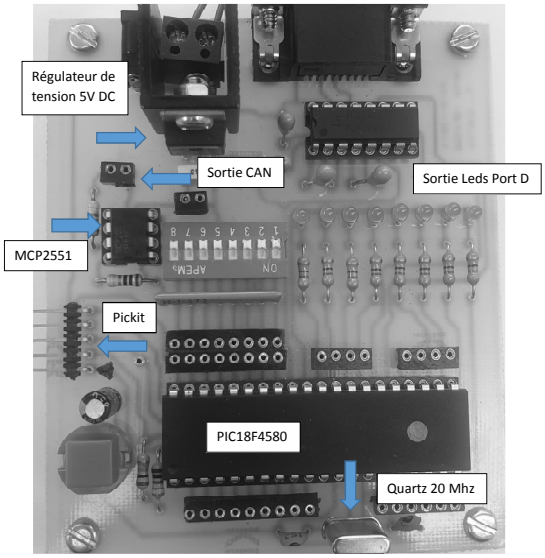
Nous avons expliqué à Mr GUSTIN les problèmes de fréquence et de rapport au temps, celui-ci nous a alors dit que pour la suite du projet cela pouvait poser problèmes et qu'il fallait résoudre ce désagrément avant de poursuivre.

L'objectif à ce moment était donc de trouver d'où venait le problème nous avons tenté de changer les condensateurs de découplage mais cela n'a eu aucun effet, par la suite nous avons changé le quartz mais une fois de plus aucun effet.

Nous avons alors décidé de mettre le microcontrôleur sur plaque labdec pour vérifier ces fonctions primaires, à ce moment le problème a été découvert, un court-circuit sur une des pates d'alimentation était à la base de nos problèmes.

Une fois le microcontrôleur changer plus de soucis sur le la fréquence de travail du microcontrôleur.

La communication pouvait ainsi se faire à 20 Mhz, ce qui nous permettaient de pouvoir communiquer avec l'autre carte.



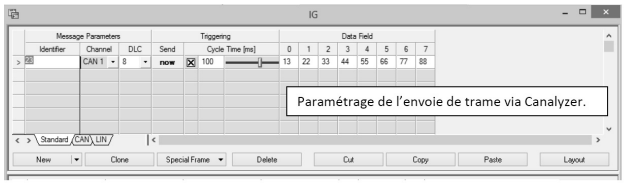
Cette carte nous permet de simuler l'envoie de trame d'un véhicule et de ne pas pouvoir passer par le Canalyzer. Cette carte émet donc les trames voulues via le MCP2551

Intégration du CAN :

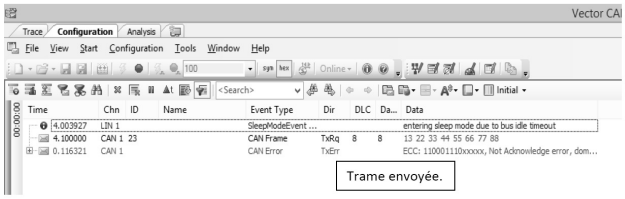
La dernière partie a intégré au système est la lecture et le traitement d'un trame CAN, celui-ci implique l'utilisation d'un module « traducteur » de CAN ici le MCP2551 utilisé pendant notre formation, celui-ci nous permet de décrypté ou de de créer une trame CAN à partir d'un microcontrôleur.

Nous avons dans un premier temps essayer de lire une trame envoyer à travers ce module, nous simulions un système CAN (voiture ou autre) avec l'ordinateur et le logiciel Canalyzer. L'idée était d'obtenir une variable avec des informations sur cette trame dans la partie envoie du programme ce qui a été fait, aujourd'hui nous sommes en capacité de recevoir un message avec une temporisation prévue les informations contenue dans cette trame et l'identifiant de la trame.

Tout ceci mis en formes à travers un message, le stockage est également possible et le programme le permet cependant nous ne possédons pas de carte SD qui nous permet de stocker celle-ci, les seuls endroits de stockage possible sont le SIM800L qui a une mémoire qui ne contient que 1 message.



Paramétrage de l'envoie de trame via Canalyzer.



Trame envoyée.

Programme final avec la réception de la donnée Can et son émission via SMS

```
#include <18F4580.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#include <can_125k.c>
#include <can_125k.h>
#use delay (clock=20000000)
#use rs232(baud=19200,xmit=PIN_C6,rcv=PIN_C7)

unsigned int buffer[8]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
int32 rx_id;
int rx_len;
struct rx_stat rxstat;
int a=26;
int c=34;
int b=13;

void main()
{
    can_init();
    while(TRUE)
    {
        if(can_kbhit())
        {
            if(can_getd(rx_id,&buffer[0],rx_len,rxstat))
            {
                if(rx_id==0x123)
                {
                    output_d(buffer[0]);
                }
            }
        }
   printf("AT+CPIN=1234");
   printf("%c",b);
   delay_ms(500);
   printf("AT+COPS?");
   printf("%c",b);
   delay_ms(500);
   printf("AT+CNUM");
   printf("%c",b);
   delay_ms(500);
   printf("AT+CMGF=1");
   printf("%c",b);
   delay_ms(500);
   printf("AT+CMGs=");
   printf("%c",c);
   printf("+33770284936");
   printf("%c",c);
   printf("%c",b);
   delay_ms(500);
tput
```

Initialisation des variables

Initialisaton du CAN

Initialisaton du CAN

Emission du SMS

```
printf("AT+CPIN=1234");
printf("%c",b);
delay_ms(500);
printf("AT+COPS?");
printf("%c",b);
delay_ms(500);
printf("AT+CNUM");
printf("%c",b);
delay_ms(500);
printf("AT+CMGF=1");
printf("%c",b);
delay_ms(500);
printf("AT+CMGs=");
printf("%c",c);
printf("+33770284936");
printf("%c",c);
printf("%c",b);
delay_ms(500);
printf("Identifiant: ");
printf("%x",rx_id);
printf("%c",b);
printf("Data: ");
printf("%x",buffer[0]);
printf("%x",buffer[1]);
printf("%x",buffer[2]);
printf("%x",buffer[3]);
printf("%x",buffer[4]);
printf("%x",buffer[5]);
printf("%x",buffer[6]);
printf("%x",buffer[7]);
delay_ms(500);
printf("%c",a);
delay_ms(500);
printf(0x1A);
delay_ms(5000);
 }
```

Emission de l'ID

Choix de l'octet a envoyé

Emission du SMS

Ce programme regroupe la réception d'une trame Can via Canalyzer ou la carte d'émission de trame. Et l'envoie du SMS en envoyant l'identifiant et la sélection de l'octet choisie (ici tous les octets sont envoyés).

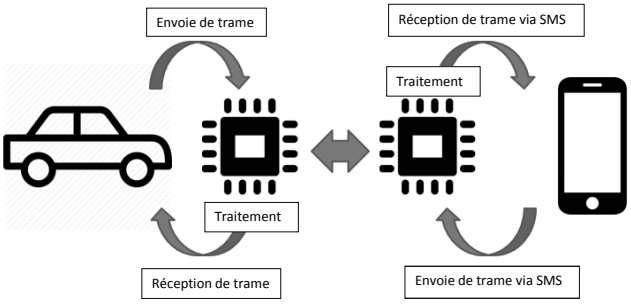
Perspective d'amélioration :

Ajout d'une deuxième carte au système permettant un stockage de données à distance et plus « sûr » sur le système.

Traduction des informations de trame en valeur appréhendable. Ne pas recevoir les octets de données mais l'état correspondant (exemple : feu allumé).

Développement d'une interface graphique sur le téléphone.

Communication via internet.



Réversibilité du système. Nous avons déjà effectué la réception de trame depuis Canalyzer et son envoie via SMS. Mais il faudrait réfléchir à envoyer une trame depuis un téléphone pour allumer des feux par exemple.

Conclusion :

Malgré les difficultés à mettre en état de fonctionnement les cartes électroniques fournies, et les retards que cela apporte sur la durée du projet, nous avons réussi à obtenir le fonctionnement basique demandée et commencé à réfléchir et travailler sur des pistes d'amélioration du projet, l'exemple étant la commande à distance de certaines trames via le téléphone dans un premier temps manuellement puis via une interface graphique.

Un autre des pistes que nous avons eu le temps de travailler mais pas finir est l'utilisation d'une deuxième carte qui servirais de relais de stockage et de communication.

Ce projet a donc été très formateur dans l'ensemble nous avons appris a surmonté certaines difficultés et compris lorsque nous n'étions pas capables de les surmontés pour demander une solution à notre professeur tuteur.

Nous avons utilisé différent modules déjà vu au cours de l'année que nous avons pu continuer à approfondir (Programmation C et Can).

Annexes :

Envoi de SMS :

```
#include <18f4580.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay (clock=20000000)
#use rs232(baud=19200,xmit=PIN_C6,rcv=PIN_C7)

int a=26;
int c=34;
int b=13;
void main ()
{
   printf("AT+CPIN=1234");
   printf("%c",b);
   delay_ms(500);
   printf("AT+COPS?");
   printf("%c",b);
   delay_ms(500);
   printf("AT+CNUM");
   printf("%c",b);
   delay_ms(500);
   printf("AT+CMGF=1");
   printf("%c",b);
   delay_ms(500);
   printf("AT+CMGs=");
   printf("%c",c);
   printf("+33770284936");
   printf("%c",c);
   printf("%c",b);
   delay_ms(500);
   printf("salut alex");
   delay_ms(500);
   printf("%c",a);
   delay_ms(500);
   printf(0x1A);
}
```

Réception SMS :

```
#include <18f4580.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay (clock=20000000)
#use rs232(baud=19200,xmit=PIN_C6,rcv=PIN_C7)

int b=13;
void main ()
{
   printf("AT+CPIN=1234");
   printf("%c",b);
   delay_ms(500);
   printf("AT+COPS?");
   printf("%c",b);
   delay_ms(500);
   printf("AT+CNUM");
   printf("%c",b);
   delay_ms(500);
   printf("AT+CMGF=1");
   printf("%c",b);
   delay_ms(500);
   printf("AT+CPMS?");
   printf("%c",b);
   printf("AT+CMGR=1");
   printf("%c",b);
   delay_ms(500);
   printf("AT+CMGD=1");
   printf("%c",b);
}
```

Envoi trame Can :

```
#include <18F4580.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#include <can_125k.c>
#include <can_125k.h>
#use delay (clock=20000000)

unsigned int variable;
unsigned int buffer[8]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

void main()
{
    can_init();
    while (TRUE)
    {
        variable=0xff;
        buffer[0]=variable;
        output_d(variable);
        can_putd(0x255,buffer,8,0,FALSE,FALSE);
        delay_ms(200);
    }
}
```

Réception trame Can :

```
#include <18F4580.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#include <can_125k.c>
#include <can_125k.h>
#use delay (clock=20000000)
#use rs232(baud=19200,xmit=PIN_C6,rcv=PIN_C7)


unsigned int buffer[8]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
int32 rx_id;
int rx_len;
struct rx_stat rxstat;


void main()
{
    can_init();
    while(TRUE)
    {

        if(can_kbhit())
        {
            if(can_getd(rx_id,&buffer[0],rx_len,rxstat))
            {
                if(rx_id==0x123)
                {
                    output_d(buffer[0]);

                }
            }
        }
    }
```

Programme émission Sms via Python

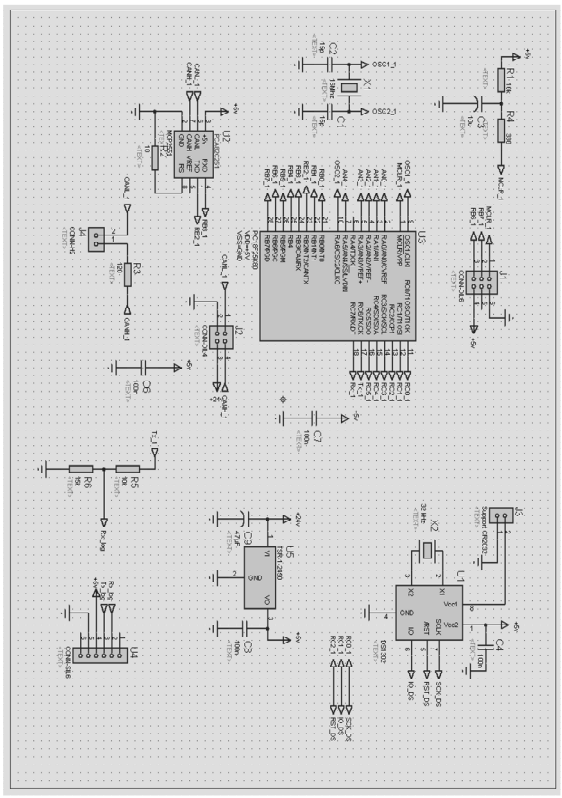
Envoi SMS Python
Depuis PC

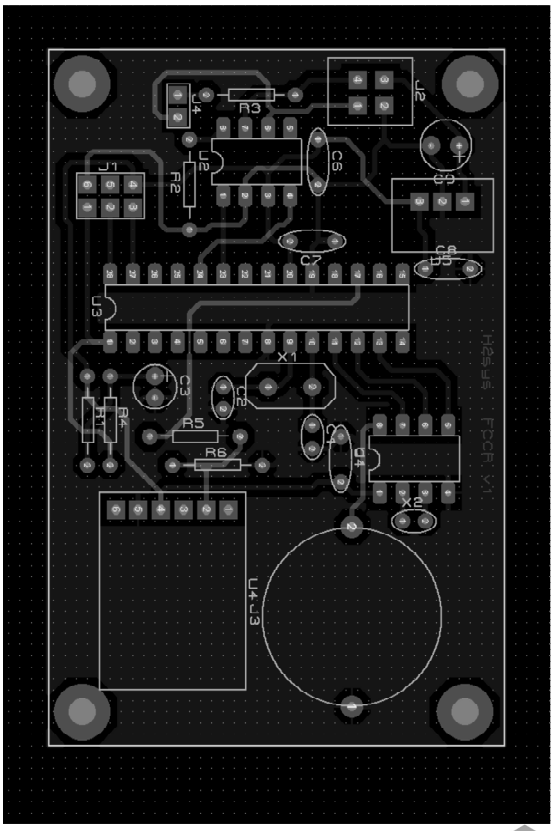
```
import serial
import time
ser = serial.Serial()
ser.close
ser.baudrate = 115200
ser.port = 2
ser.open()
ser.timeout = 0.100
code_pin=0

for i in range(10):
    a=ser.readline()
    if a!='':
        print(a)
ser.write( "AT"+chr(13))
for i in range(10):
    a=ser.readline()
    if a!='':
        print(a)
ser.write("AT+CPIN?"+chr(13))
for i in range(5):
    a=ser.readline()
    if a!='':
        print("ligne ",i," ",a)
    if a=="+CPIN: READY\r\n":
        code_pin=1
if code_pin==0:
    ser.write("AT+CPIN="+chr(34)+"1234"+chr(34)+chr(13))
    for i in range(5):
        a=ser.readline()
        if a!='':
            print("ligne ",i," ",a)
# passage mode text
ser.write("AT+CMGF=1"+chr(13))
for i in range(5):
    a=ser.readline()
    if a!='':
        print("ligne ",i," ",a)
#SMS
ser.write("AT+CMGS="+chr(34)+"0672939919"+chr(34)+chr(13))
for i in range(5):
        a=ser.readline()
        if a!='':
            print("ligne ",i," ",a)
ser.write("Message test"+chr(13))
for i in range(5):
        a=ser.readline()
        if a!='':
            print("ligne ",i," ",a)
#CTRL+L code 1A en Hexa
ser.write(chr(26))
for i in range(5):
        a=ser.readline()
        if a!='':
            print("ligne ",i," ",a)

ser.close
```

GEII
Département Génie Électrique
& Informatique Industrielle
IUT Belfort-Montbéliard

Recept 02.py

```
import serial
import time
ser = serial.Serial()
ser.close
ser.baudrate = 115200
ser.port = 2
ser.open()
ser.timeout = 0.100
code_pin=0
for i in range(10):
    a=ser.readline()
    if a!='':
        print(a)
ser.write( "AT"+chr(13))
for i in range(10):
    a=ser.readline()
    if a!='':
        print(a)
ser.write("AT+CPIN?"+chr(13))
for i in range(5):
    a=ser.readline()
    if a!='':
        print("ligne ",i," ",a)
    if a=="+CPIN: READY\r\n":
        code_pin=1
if code_pin==0:
    ser.write("AT+CPIN="+chr(34)+"1234"+chr(34)+chr(13))
    for i in range(5):
        a=ser.readline()
        if a!='':
            print("ligne ",i," ",a)
# passage mode text
ser.write("AT+CMGF=1"+chr(13))
for i in range(5):
    a=ser.readline()
    if a!='':
        print("ligne ",i," ",a)
# test reception nb SMS
ser.write("AT+CPMS?"+chr(13))
tab=[]
for i in range(6):
    a=ser.readline()
    if a!='':
        print("ligne ",i," ",a)
    #ligne 1 : info nb SMS
    if i==1:
        tab=a.split(',')
        nb_sms=int(tab[1])
print ('nb_sms ',nb_sms)
#lecture des derniers sms et effacement
num_sms=1
while num_sms<=nb_sms:
   ser.write("AT+CMGR="+str(num_sms)+chr(13))
   for i in range(5):
       a=ser.readline()
       if (a!='') and (i==2):
           print(a)
   #effacement
   ser.write("AT+CMGD="+str(num_sms)+chr(13))
   for i in range(6):
    a=ser.readline()
    if a!='':
        print("ligne ",i," ",a)
   num_sms=num_sms+1
ser.close
```




J4
R3
J2
J1
L2
C6
R2
C9
C7
C8
D5
U3
X1
C3
C2
R4
R1
R5
R6
C4
D4
X2
U+C13