

Rapport 09/07/2021


Création d'un GANTT

GANTT CATCOLLARBOWL

	Semaine 1	Semaine 2	Semaine 3	Semaine 4	Semaine 5	Semaine 6
[Etude] Connexion Wifi RPI en C						
[Etude] Comparatif API SMS						
[Dev] envoi SMS via API						
[Dev] Contrôle moteur pas à pas						
[Dev] Traitement SMS MKR1400						
[Dev] Interface GTK heure						
[Fab] Prototypage V1						
[Dev] Raccorder les fonctions ensemble						
[Fab] Raccorder le proto au moteur PàP						
[Etude] Localisation MKR1400						
[Conception/réalisation] Echange d'informations de localisation						

Légende : Fabien Julien Travail à deux

Définition de la répartition des tâches et de leur durée.

On a divisé les tâches ci-dessus en sous-tâches afin de pouvoir estimer au mieux la durée nécessaire pour les réaliser (cf document  todo)

Nous nous les sommes réparties sur [Trello](#).

Comparatif des différents APIs d'envoi de SMS

nom	Prix / sms (pack prépayés)	Avantages	Inconvénient
SMSFactor	0.065€	<ul style="list-style-type: none">- 10 SMS de test gratuits (renouvelables en illimité)- documentation en C très fournie- pas d'engagement- aucune date limite d'utilisation- requête HTTP	<ul style="list-style-type: none">- pas d'envoi d SMS en différé
smsmode	0.0906€	<ul style="list-style-type: none">- aucune date limite d'utilisation- requête HTTP	<ul style="list-style-type: none">- pas de documentation en C- documentation en PDF- pas d'envoi de SMS en différé- Le plus cher- propose des abonnements
sms partner	0.045€	<ul style="list-style-type: none">- 20 SMS de test gratuit- documentation très fournis- le moins cher- envoi instantané ou différé- aucune date limite d'utilisation- tests dans un mode sandbox- pas d'engagement- requête HTTP	<ul style="list-style-type: none">- pas de documentation en C

Décision prise : test avec SMS partner dans un premier temps, afin de savoir si le mode sandbox comptabilise les SMS, afin d'utiliser l'API pour nos tests. En second choix, SMSFactor semble être une bonne solution pour développer notre prototype, avec ses SMS offerts "à volonté", et une bonne doc en C.

Matériel

Nous avons eu un appel avec Fred afin de nous parler du matériel qu'il a déposé à Centrale. On a tout récupéré ce vendredi 09/07 auprès de F. Verbrughe.

Détail du matériel

- Un écran LCD
- Un MKR1400
- Une batterie 3.7V
- Un moteur pas à pas
- Un raspberry pi
- Contacteur (comme sur les fenêtres)
- Diodes avec un aimant
- Des câbles
- Une alimentation secteur
- Un vibreur (ou buzzer)
- Un explorer HAT pro
- Grove pi

Prochaine semaine :

Début du développement chacun de notre côté, mise en place de l'api SMS et de l'interface du projet.

Contactez notre opérateur car il faudrait apparemment un forfait M2M.

Rapport 16/07/2021

Début du développement de la mise en place de l'api SMS.

Mise en place de la librairie cJSON, afin de communiquer avec l'API via cURL

```
/mnt/d/dev/C/sms_api ./build/test.o
{
    "apiKey": [REDACTED],
    "phoneNumbers": [REDACTED],
    "message": "Hello World !",
    "sender": "Hypario"
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// #include <curl/curl.h>
#include <cjson/cJSON.h>

int main(void)
{
    char token[] = [REDACTED];
    char numbers[] = [REDACTED];
    char message[] = "Hello World !";
    char sender[] = "Hypario";

    char *out;

    cJSON *postdata;

    // CURLcode resp;

    postdata = cJSON_CreateObject();

    cJSON_AddItemToObject(postdata, "apiKey", cJSON_CreateString(token));
    cJSON_AddItemToObject(postdata, "phoneNumbers", cJSON_CreateString(numbers));
    cJSON_AddItemToObject(postdata, "message", cJSON_CreateString(message));
    cJSON_AddItemToObject(postdata, "sender", cJSON_CreateString(sender));

    out = cJSON_Print(postdata);
    printf("%s\n", out);
}
```

Il a fallu cependant installer les librairies, pour cela un script bash a été mis en place afin de télécharger et installer tous les requirements.

Malheureusement, beaucoup d'informations sont affichées lors de l'utilisation du script, cela n'est pas gênant et pourra être modifié un peu plus tard lors du développement.

```

1  #!/usr/bin/env bash
2
3  sudo apt update
4  sudo apt install -y libcurl4-openssl-dev
5
6  mkdir -p libs_src
7
8  curdir=$(pwd)
9
10 # téléchargement de la librairie cJSON
11 echo "téléchargement des sources de la librairie cJSON"
12 if [ ! -d libs_src/cJSON ]; then
13     wget "https://github.com/DaveGamble/cJSON/archive/refs/tags/v1.7.13.zip" -O libs_src/cJSON.zip
14
15     # dépaquetage
16     unzip libs_src/cJSON.zip -d libs_src
17     mv libs_src/cJSON-1.7.13 libs_src/cJSON
18
19     rm -rf libs_src/cJSON.zip
20 else
21     echo "source de la librairie cJSON présente"
22 fi
23
24 echo $curdir
25
26 # compilation de cJSON
27 echo "compilation et installation de cJSON"
28 if [ ! -d libs/cJSON ]; then
29     mkdir -p libs/cJSON
30     pushd libs_src/cJSON
31     make all
32     sudo make PREFIX=$curdir/libs/cJSON install
33     sudo cp -a $curdir/libs/cJSON/lib/* /usr/lib
34     sudo ldconfig -n -v /usr/lib
35     popd
36     echo "compilation de cJSON terminé"
37 else
38     echo "cJSON déjà compilé"
39 fi
40
41 echo "Merci d'utiliser la commande make"

```

Ensuite, mise en place de l'envoi de la requête cURL.

```
out = cJSON_Print(postdata);
printf("%s\n", out);

CURL *curl = curl_easy_init();

if (curl) {
    curl_easy_setopt(curl, CURLOPT_URL, "https://api.smspartner.fr/v1/send");
    curl_easy_setopt(curl, CURLOPT_TIMEOUT, 10);
    curl_easy_setopt(curl, CURLOPT_POST, 1);
    curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, (long) strlen(out));
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, out);

    resp = curl_easy_perform(curl);

    if (resp != CURLE_OK) {
        fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(resp));
    }

    curl_easy_cleanup(curl);
}

cJSON_Delete(postdata);

return 0;
```

TODO : retenter après un certain temps

Rapport 23/07/2021

SMS partner ne laissant aucun accès apparent à une sandbox pour effectuer des tests, adaptation du code pour SMSFactor afin de tester le code.

Après adaptation du code, nous avons bien 10 SMS gratuits nous permettant de tester l'application.

```
cJSON *postdata, *sms, *recipients, *car, *message, *gsm;

char src[200]; // header for authorization
struct curl_slist *headers = NULL;
CURLcode resp;

postdata = cJSON_CreateObject();

cJSON_AddItemToObject(postdata, "sms", sms = cJSON_CreateObject());
cJSON_AddItemToObject(sms, "recipients", recipients = cJSON_CreateObject());
cJSON_AddItemToObject(sms, "message", message = cJSON_CreateObject());
cJSON_AddItemToObject(recipients, "gsm", gsm = cJSON_CreateArray());

int i = 0;
for(i = 0 ; i < sizeof(numbers)/sizeof(char*) ; i++){
    cJSON_AddItemToArray(gsm, car = cJSON_CreateObject());
    cJSON_AddItemToObject(car, "value", cJSON_CreateString(numbers[i]));
}
cJSON_AddItemToObject(message, "text", cJSON_CreateString(data));

out = cJSON_Print(postdata);
printf("%s\n", out);

// préparation requête CURL

CURL *curl = curl_easy_init();

strcpy(src, "Authorization: Bearer ");
strcat(src, token);

headers = curl_slist_append(headers, "Content-Type: application/json");
headers = curl_slist_append(headers, "Accept: application/json");
headers = curl_slist_append(headers, src);

if(curl){
    curl_easy_setopt(curl, CURLOPT_URL, "https://api.smsfactor.com/send");
    curl_easy_setopt(curl, CURLOPT_POST, 1);
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
    curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, (long) strlen(out));
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, out);
    resp = curl_easy_perform(curl);

    if (resp != CURLE_OK) {
        fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(resp));
    }
    curl_easy_cleanup(curl);
}
```

Les messages sont bien reçus sur le téléphone, le code est donc fonctionnel.

```
/mnt/d/dev/C/sms_api ./build/test.o
{
  "sms": {
    "recipients": {
      "gsm": [{
        "value": "33645877307"
      }],
      "message": {
        "text": "Hello World !"
      }
    }
  }
}
{"status":1,"message":"OK","ticket":"96593374","cost":1,"credits":9,"total":1,"sent":1,"blacklisted":0,"duplicated":0,"invalid":0,"npai":0}
```

Celui-ci me retourne combien de crédit il nous reste, combien de messages ont été envoyés, si les messages se sont bien envoyés.

Il reste encore à traiter le message reçu sur la RPI et définir un protocole de communication.

Rapport 30/07/2021

Julien ayant eu le covid, nous n'avons pas eu l'occasion de nous voir, je n'ai eu aucun matériel, j'en ai donc profité pour me renseigner sur le moteur pas à pas et quel librairie GPIO utiliser (wiringPi ou pigpio)

Sur les deux pages suivantes vous trouverez une utilisation du moteur pas à pas en python, afin de comprendre le fonctionnement de celui-ci.


```
from time import sleep
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

# Initialisation des variables
A=18
B=23
C=24
D=25
time = 0.001

# Definition des broches
GPIO.setup(A,GPIO.OUT)
GPIO.setup(B,GPIO.OUT)
GPIO.setup(C,GPIO.OUT)
GPIO.setup(D,GPIO.OUT)
GPIO.output(A, False)
GPIO.output(B, False)
GPIO.output(C, False)
GPIO.output(D, False)

# Definition des etapes des moteurs
def Step1():
    GPIO.output(D, True)
    sleep (time)
    GPIO.output(D, False)

def Step2():
    GPIO.output(D, True)
    GPIO.output(C, True)
    sleep (time)
    GPIO.output(D, False)
    GPIO.output(C, False)
```

```

def Step3():
    GPIO.output(C, True)
    sleep (time)
    GPIO.output(C, False)

def Step4():
    GPIO.output(B, True)
    GPIO.output(C, True)
    sleep (time)
    GPIO.output(B, False)
    GPIO.output(C, False)

def Step5():
    GPIO.output(B, True)
    sleep (time)
    GPIO.output(B, False)

def Step6():
    GPIO.output(A, True)
    GPIO.output(B, True)
    sleep (time)
    GPIO.output(A, False)
    GPIO.output(B, False)

def Step7():
    GPIO.output(A, True)
    sleep (time)
    GPIO.output(A, False)

def Step8():
    GPIO.output(D, True)
    GPIO.output(A, True)
    sleep (time)
    GPIO.output(D, False)
    GPIO.output(A, False)

# Programme pour un tour complet
for i in range (512):
    Step1()
    Step2()
    Step3()
    Step4()
    Step5()
    Step6()
    Step7()
    Step8()

GPIO.cleanup()

```

Le moteur fait un tour complet après 4096 pas. Pour faire 8 pas, il faut activer successivement les pins 25, 24, 23 et 18 (dans cet ordre).

Il faut laisser un temps de 1ms entre chaque pin afin de laisser le moteur tourner.

documentation du moteur pas à pas :

<https://www.gotronic.fr/pj2-sbc-moto1-fr-1519.pdf>

Rapport 06/08/2021

Nous avons pu nous voir, ainsi Julien a pu me donner le matériel pour le moteur pas à pas, malheureusement je ne pouvais pas m'en servir n'ayant aucun câble pour le brancher puisqu'il nous fallait des câbles femelles et nous n'avions que des câbles mâles.

Rapport 13/08/2021

J'ai pu récupérer des câbles femelles grâce à mon entreprise et effectuer les tests du code en C.

Malheureusement le moteur ne tournait pas avec la librairie pigpio, on ne sentait le moteur tourner qu'avec un temps d'attente de 1 seconde, en dessous, le moteur ne tournait pas.

Rapport 20/08/2021

Le moteur tourne enfin, nous utilisons maintenant wiringPi et la fonction delay de la librairie stdlib qui prends en paramètre des millisecondes plutôt que des secondes, nous pensons que le problème venait de la fonction sleep de la librairie unistd qui ne permettait pas de descendre à la milliseconde et empêcher le moteur de tourner correctement.