RAPPORT DE SEANCE :

Séance n°4 - 19 janvier 2023

Combiner Capacitive Sensing et Restitution du son :

La séance d'aujourd'hui consistait à réussir à restituer du son lorsque nous touchions les capsules. Pour celà, nous avons combiner notre travail sur le Capacitive Sensing, effectué avant les vacances, et sur la restitution du son. Clarisse s'est occupée du montage pendant que je me chargait du code.

Pour le code, je me suis inspirée des exemples de code trouvés dans les deux bibliothèques « CapacitiveSensor.h » et « SoftwareSerial.h » et j'ai fini par obtenir le code ci-dessous :

(certaines lignes de codes sont déjà expliquées en commentaire)

```
#include <CapacitiveSensor.h>

#define COMMON_PIN 53

CapacitiveSensor   first = CapacitiveSensor(53,51);
CapacitiveSensor   second = CapacitiveSensor(53,50);


#include <SoftwareSerial.h>

#define CMD_PLAY_NEXT 0x01
#define CMD_PLAY_PREV 0x02
#define CMD_PLAY_W_INDEX 0x03
#define CMD_SET_VOLUME 0x06
#define CMD_SEL_DEV 0x09
#define CMD_PLAY_W_VOL 0x22
#define CMD_PLAY 0x0D
#define CMD_PAUSE 0x0E
#define CMD_SINGLE_CYCLE 0x19
#define CMD_PLAY_WITH_FOLDER 0x0F

#define DEV_TF 0x02
#define SINGLE_CYCLE_ON 0x00
#define SINGLE_CYCLE_OFF 0x01

#define ARDUINO_RX 7  // Arduino Pin connected to the TX of the Serial MP3 Player module
#define ARDUINO_TX 6  // Arduino Pin connected to the RX of the Serial MP3 Player module

SoftwareSerial mp3(ARDUINO_RX, ARDUINO_TX);

int selected_sound = 0;
```

Nous avons pour le moment testé le code pour seulement 2 capsules, une reliée à l'entrée 51 et l'autre à l'entrée 50. Le pin 53 joue le rôle de « capteur » pour chacune des capsules (qui sont toutes reliées entre elles par un même câble).

```
void setup()
{
  Serial.begin(9600);
  mp3.begin(9600);
  delay(500);  // wait chip initialization is complete

  mp3_command(CMD_SEL_DEV, DEV_TF);  // select the TF card
  delay(200);

  mp3_command(CMD_SET_VOLUME,50);

  //mp3_command(CMD_PLAY_WITH_FOLDER, 0x0F00102);       // Play mp3
  // mp3_command(CMD_PAUSE, 0x0000);      // Pause mp3
  //mp3_command(CMD_PLAY_NEXT, 0x0000);  // Play next mp3
  //mp3_command(CMD_PLAY_PREV, 0x0000);  // Play previous mp3
  //mp3_command(CMD_SET_VOLUME, 30);     // Change volume to 30
}
```

```
void loop()
{
  long start = millis();
  long total1 =  first.capacitiveSensor(30);
  long total2 =  second.capacitiveSensor(30);

  Serial.print(millis() - start);        // check on performance in milliseconds
  Serial.print("\t");                    // tab character for debug windown spacing

  Serial.print(total1);
  Serial.print("\t");
  Serial.println(total2);

  if(total1 > 500 and total2 > 500) {
    selected_sound = 0;
    mp3_command(CMD_PAUSE, 0x0000);
  } else if(total1 > 500) {
    if(selected_sound != 1) {
      selected_sound = 1;
      mp3_command(CMD_PLAY_WITH_FOLDER, 0x0F00101);
    }
  } else if(total2 > 500) {
    if(selected_sound != 2) {
      selected_sound = 2;
      mp3_command(CMD_PLAY_WITH_FOLDER, 0x0F00102);
    }
  }
  delay(10);                             // arbitrary delay to limit data to serial port
}
```

Cela nous permettait de vérifiais le fonctionnement du Capacitive Sensing au cours des modification du montage et des soudures

Met en pause le son lorsque les deux capsules sont touchées

Joue le fichier 001.mp3 du dossier 001 lorsque la première capsule est touchée

Joue le fichier 002.mp3 du dossier 001 lorsque la première capsule est touchée

```
void mp3_command(int8_t command, int16_t dat) {
  int8_t frame[8] = { 0 };
  frame[0] = 0x7e;                // starting byte
  frame[1] = 0xff;                // version
  frame[2] = 0x06;                // the number of bytes of the command without starting byte and en
  frame[3] = command;             //
  frame[4] = 0x00;                // 0x00 = no feedback, 0x01 = feedback
  frame[5] = (int8_t)(dat >> 8);  // data high byte
  frame[6] = (int8_t)(dat);       // data low byte
  frame[7] = 0xef;                // ending byte
  for (uint8_t i = 0; i < 8; i++) {
    mp3.write(frame[i]);
  }
}
```

Il a fallu d'abord tester le fonctionnement avec les sons procurés par le professeur lorsque la carte SD nous a été prêtée, puis, une fois nos 16 fichiers sons trouvés pour nos 16 notes de piano différentes, nous avons pu les ajouter dans la carte SD.

Le code sera perfectionné la semaine prochaine en utilisant de listes et de boucles « for » une fois toutes les autres capsules ajoutées.

Soudure :

J'ai également passé les 45 dernière minutes de la séance à tenter de souder les résistances aux capsules restantes (et cela est toujours aussi compliqué, l'étain tenant très difficilement sur les capsules).