Robot Léo

15/06/2021

—

HAMON Dimitri - LE GAC Pierre - MARU Théodros - ROUILLÉ Lucas

IUT Nantes
Module ER2
Carquefou - 44470

Sommaire

Introduction :

Pendant ce semestre, nous avons pu étudier les différents points importants liés aux projets lors du module CP2. Le but du module ER2 est de mettre en pratique ce qu'on a pu étudier lors de celui en CP2. On va devoir travailler en groupe, chacun sur une fonction du projet. On devra gérer notre temps en fonction du planning, prendre en compte les risques afin de mieux s'y préparer et de les éviter et prendre en compte les aléas. Il faudra aussi prendre en compte le budget. Même si le financement du projet est couvert par l'IUT, il est très intéressant de se comporter en condition réelle (comme en entreprise) et donc d'éviter un maximum de coûts inutiles. L'objectif sera aussi de faire appel le moins possible aux différents professeurs que l'on pourrait comparer à des tuteurs même s'ils sont présents essentiellement pour nous guider, on est une classe entière alors pendant que le professeur est occupé, on peut imaginer que notre tuteur le serait également en condition réelle. En clair, nous allons devoir nous comporter de la même manière que si nous étions en entreprise. Nous allons donc pour la première fois participer à un projet qui a pour nom : " Robot Léo".

Risques

Dans un projet, il y aura toujours des risques, mais le but va être de leur donner un degré de probabilité et un degré de gravité.

On commence par une liste des risques :

- Erreur dans le schéma. (1)
- Erreur dans le routage. (2)
- Mauvaise soudure. (3)
- Mauvaise polarisation d'un composant. (4)
- Surtension lors d'une mise en tension. (5)
- Surcharge des super-condensateurs. (6)
- Chute d'une carte (maladresse). (7)
- Matériel de mesure non fonctionnel. (8)

Maintenant que ceci a été fait, on va les ranger dans un tableau et les trier :

		Gravité		
		Faible	Modérée	Élevé
Probabilité du risque	Faible		7	5 et 6
	Modérée	8	4	1 et 2
	Élevé		3	

On remarque que le nombre de risques est relativement faible et que leurs gravités sont minimes. Le but est de prendre en compte les risques et de pouvoir les incrémenter dans l'emploi du temps. On appelle ça une marge, cependant, il ne faut pas oublier les aléas (des risques imprévus, mais qui ont été pris en compte dans l'emploi du temps).

Pour ce qui est de la classification, certains risques sont rangés dans "Élevé" alors que dans un projet traditionnel, ils seraient d'une gravité modérée, voire faible, car la durée du projet est très courte (44 h).

.

Budget :

Dans cette partie, on va essayer d'estimer le coût de fabrication de chaque carte, le but de ces deux parties budget en début et fin de projet, c'est de voir les estimations du budget par rapport à un vrai projet.

On ne tiendra pas compte du fait qu'il y a différentes valeurs selon la résistance ni pour les condensateurs. On est conscient que les prix unitaires ont ce prix, car ils sont commandés en lots.

Tout d'abord, on fait un listing des composants nécessaires à la conception (prix unitaire sur RS Components FR) :

- Résistances CMS x18 :

On prend une marge de 2 : 0.29 € sans les frais de livraison.

- condensateurs 100µF :

On prend une marge de 2 : 0.58 € sans les frais de livraison.

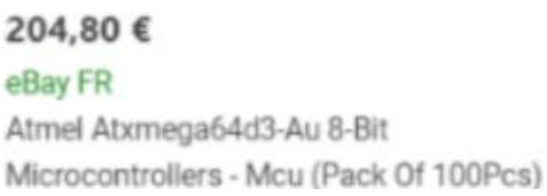
- condensateurs 1000µF :

On prend une marge de 2 : 1.47 € sans les frais de livraison.

- Port USB x3 :

On prend 1 port de plus de marge : 8.40€ sans les frais de livraison.

- Samd21 x3 (prix venant d'eBay) :



On prend une marge de 2 : 4.096 € sans les frais de livraison.

- DTMF x1:

1,167 €
HT

1,40 €
TTC

On prend une marge de 2 : 2.8€ sans les frais de livraison.

- Qwartz x3 :

0,216 €
HT

0,259 €
TTC

On ne prend pas de marge : 0.78€ sans les frais de livraison.

- Super-condensateur x2 :

4,097 €
HT

4,916 €
TTC

On ne prend pas de marge : 10 € sans les frais de livraison.

- Bornier x11 :

2,65 € Digi-Key France

On ne prend pas de marge : 29.15€ (à mettre entre guillemets, car il y a des borniers très différents entre la fp2 et les autres fonctions) sans les frais de livraison.

On négligera le prix des autres composants, soit prix totales des composants = 29.15+10+0.78+2.8+6.12+8.4+1.62+1.47+0.29 = 60.63€

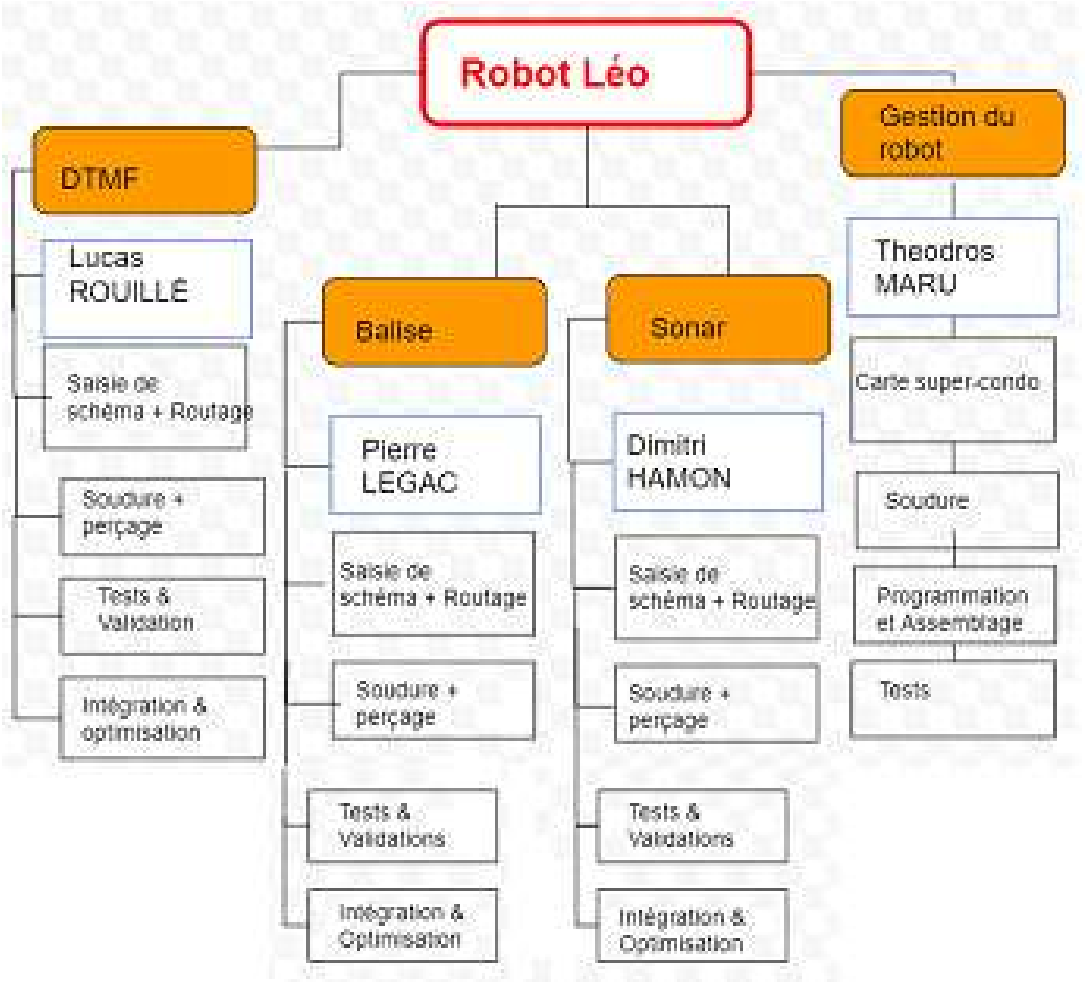
On prendra une main d'œuvre de 4 employés pour 13.33€/h Net (sous 35h semaine ceci représente 1866/mois ce qui est cohérent pour un technicien). On doit réaliser le projet en 11 séances de 4h30 soit 660€ sans compter les charges de l'entreprise.

On ne comptera pas le coût de la création des cartes, machine CMS, fer à souder, locaux (location + électricité, etc).

On prendra une marge budgétaire de l'ordre de 40%, soit le coût du budget est estimé à (660+60.63)*1.4 = 1008.9 €

Work Package

On a analysé les risques et le budget désormais on passe à la dernière étape avant de pouvoir commencer le projet le “work package”.

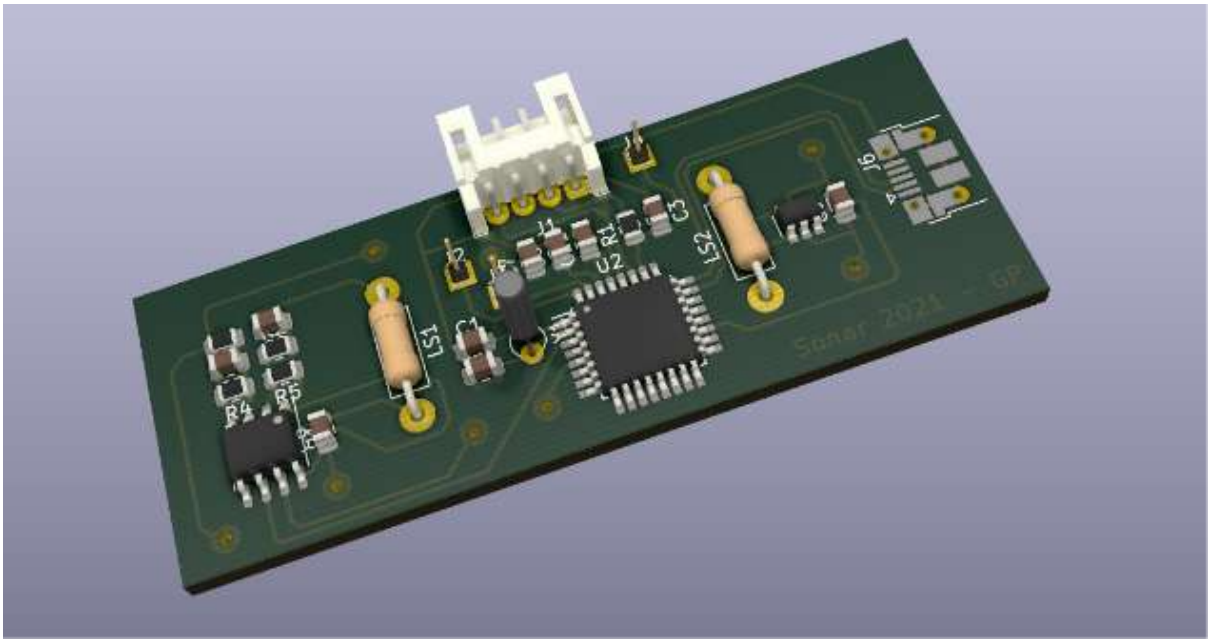


On peut remarquer que les 3 règles du “Work Package” ont été respectées :

- Résultat (nom de la fonction).
- Responsable (nom de la personne qui s’occupe de la fonction).
- Ressources (nom des étapes qui conduisent au résultat)

La fonction FP1 « détection d'obstacles »

Pierre LE GAC



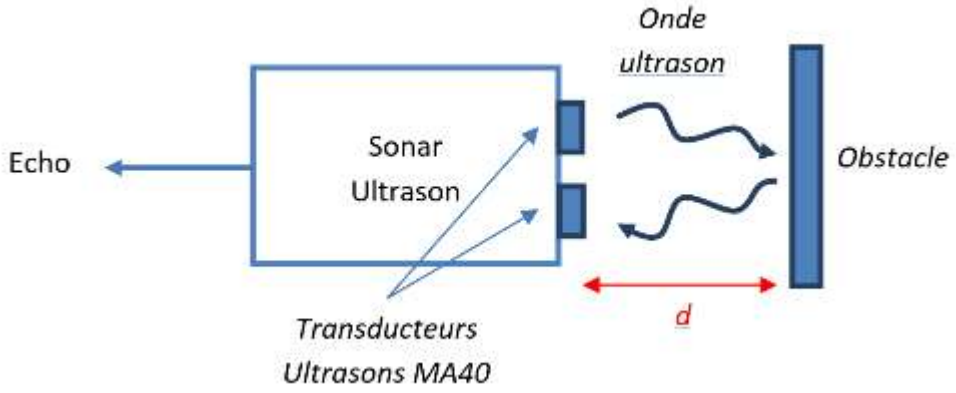
1- Introduction

Lors de son déplacement sur sa cible, le robot Léo doit pouvoir éviter divers obstacles sur le parcours. On trouvera deux types d'obstacles :

- Les obstacles mobiles correspondant aux robots adverses
- Les obstacles fixes disposés sur le parcours

Le détecteur doit être capable de mesurer une distance minimum de 50cms entre le robot et l'obstacle (fixe ou mobile) avec une précision de 1cm.

Pour réaliser le détecteur, nous utiliserons le principe du sonar ultrason. Celui-ci repose sur l'émission et la réception d'une onde ultrasonique à 40kHz (burst de 250us soit 10 périodes à 40kHz) via deux transducteurs tel que le montre la figure ci-dessous :



Comme une onde se propage à la vitesse de 330m/s, on peut facilement calculer la distance d (en mètre) séparant le sonar de l'obstacle avec la formule : d = 330 * t/ 2 (car il y a l'aller et le retour)

Répartition des tâches:

Le projet se déroule sur 11 séances de BE. Le planning prévisionnel pour la fonction FP1 est indiqué ci-dessous :

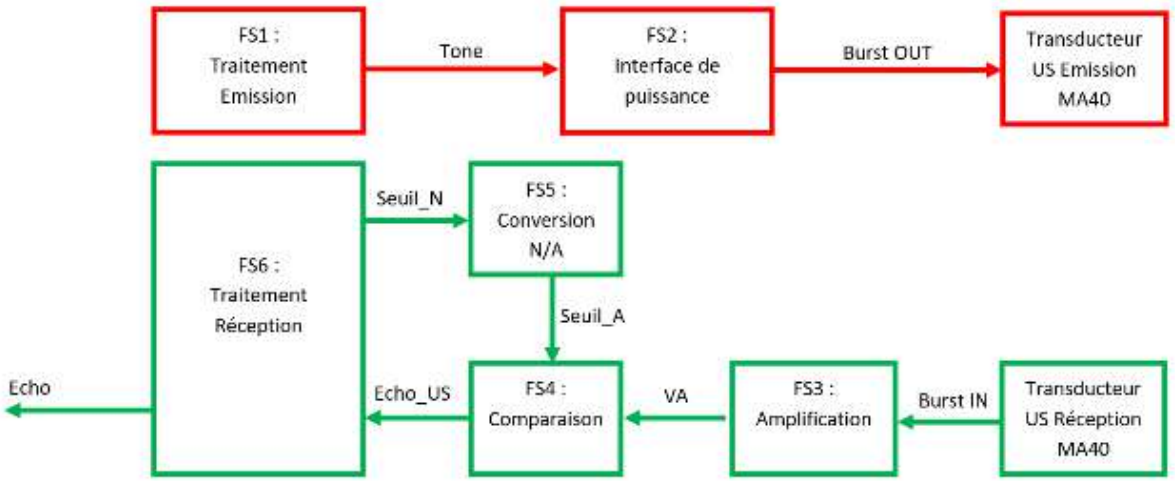
Séances	01	02	03	04	05	06	07	08	09	10	11
Etude	h40	h30									
Conception			h30	h30	h30						
Fabrication						h30	h30				
Test								h30	h30		
Intégration										h30	h30

Phase d'étude : analyse théorique du schéma
Phase de conception : CAO sous kicad (affectation des empreintes, routage)
Phase de fabrication : perçage, soudure des composants
Phase de test : validation du bon fonctionnement du sonar (utilisation d'un banc de

test)

Phase d'intégration : assemblage des cartes électroniques sur le robot et validation de la stratégie

2- Description fonctionnelle



- Fonction FS1 : Traitement Emission

La fonction « traitement émission » assure la production du signal électrique Tone à transmettre au transducteur US d'émission toutes les 20ms **environ** (cadence de mesure).

Sortie Tone : train d'impulsions de durée t
Amplitude : 0V au repos et 0V/+3.3V à l'état actif.
Durée à l'état actif : t = 250us (10 périodes ultrason à 40kHz)
Période : transmission toutes les 20ms

- ## Fonction FS2 : Interface de puissance

La fonction « Interface de puissance » assure l'adaptation des niveaux électriques (courant et tension) entre la sortie de la fonction « Traitement Emission » et le transducteur US.

Entrées Tone : train d'impulsions de durée t
Amplitude : 0V au repos et 0V/+3.3V à l'état actif.
Durée à l'état actif : t = 250us (10 périodes ultrason à 40kHz)
Période : transmission toutes les 20ms

Sortie Burst_OUT : train d'impulsions de durée t
Amplitude : +3.3V au repos et -3.3V/+3.3V à l'état actif
Durée à l'état actif : t = 250us (10 périodes ultrason à 40kHz)
Période : transmission toutes les 20ms

- ## Fonction FS3 : Amplification

Cette fonction assure l'amplification de la tension renvoyée par le transducteur ultrason de réception.

Entrée Burst IN : signal ultrason de faible amplitude
Sortie VA : signal ultrason amplifié

Plus l'obstacle est éloigné et plus le signal de réception est faible. On utilise donc un amplificateur à 1 AOP inverseur en mono-tension .

- ## Fonction FS4 : Comparaison

Cette fonction assure la mise en forme du signal ultrason amplifié afin de pouvoir le renvoyer à la fonction « traitement réception».

Entrée VA : signal ultrason amplifié
Seuil_A : seuil de comparaison analogique (valeur comprise entre 0 et 3.3V)

Sortie Echo_US : Signal de retour ultrason
Amplitude : 0/3.3V

- ## Fonction FS5 : Conversion numérique analogique

Cette fonction produit un seuil de comparaison pour la fonction « comparaison » (programmable par la fonction « traitement). L'amplitude de ce seuil dépendra de la sensibilité souhaitée.

Entrée Seuil_N : seuil de comparaison numérique
Codage sur 8 bits (valeur variant de 0 à 255)
Sortie Seuil_A : seuil de comparaison analogique
Valeur variant de 0 à 3.3V.

- ## Fonction FS6 : Traitement Réception

La fonction « traitement réception » assure la mise en forme du signal echo afin de le transmettre à la fonction FP1 « gestion du robot ». Elle assure également la production d'un seuil numérique pour la fonction « comparaison ».

Entrées Echo_US : Signal de retour ultrason
Amplitude : 0/3.3V

Sortie Echo : Impulsion de durée t
Amplitude : 0/3.3V
Durée à l'état actif (état haut) : de 0 à TIMEOUT
Seuil_N : valeur numérique du seuil de comparaison
Codage sur 8 bits (valeur variant de 0 à 255)

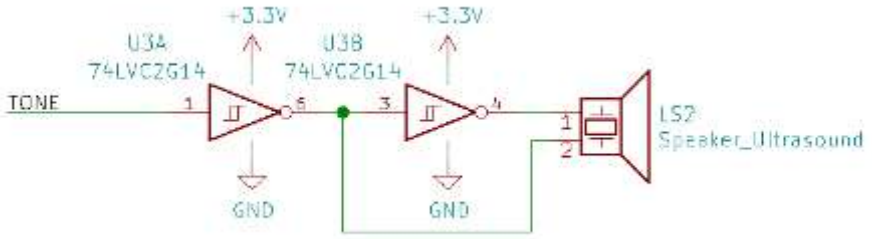
Si écho_US = 1 alors il y a un obstacle et écho = 0 Si écho_US = 0 alors il n'y a pas d'obstacle et écho = 1

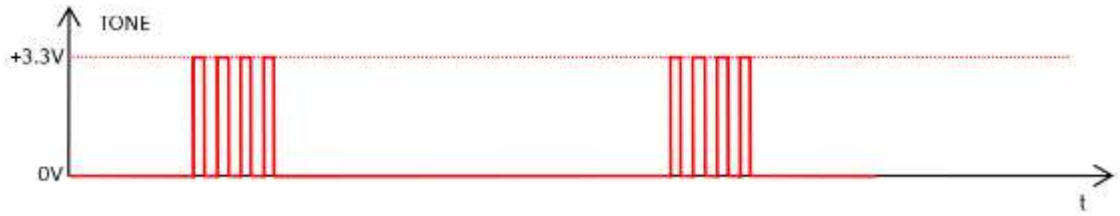
3- Etude théorique Gestion de l'émission ultrason

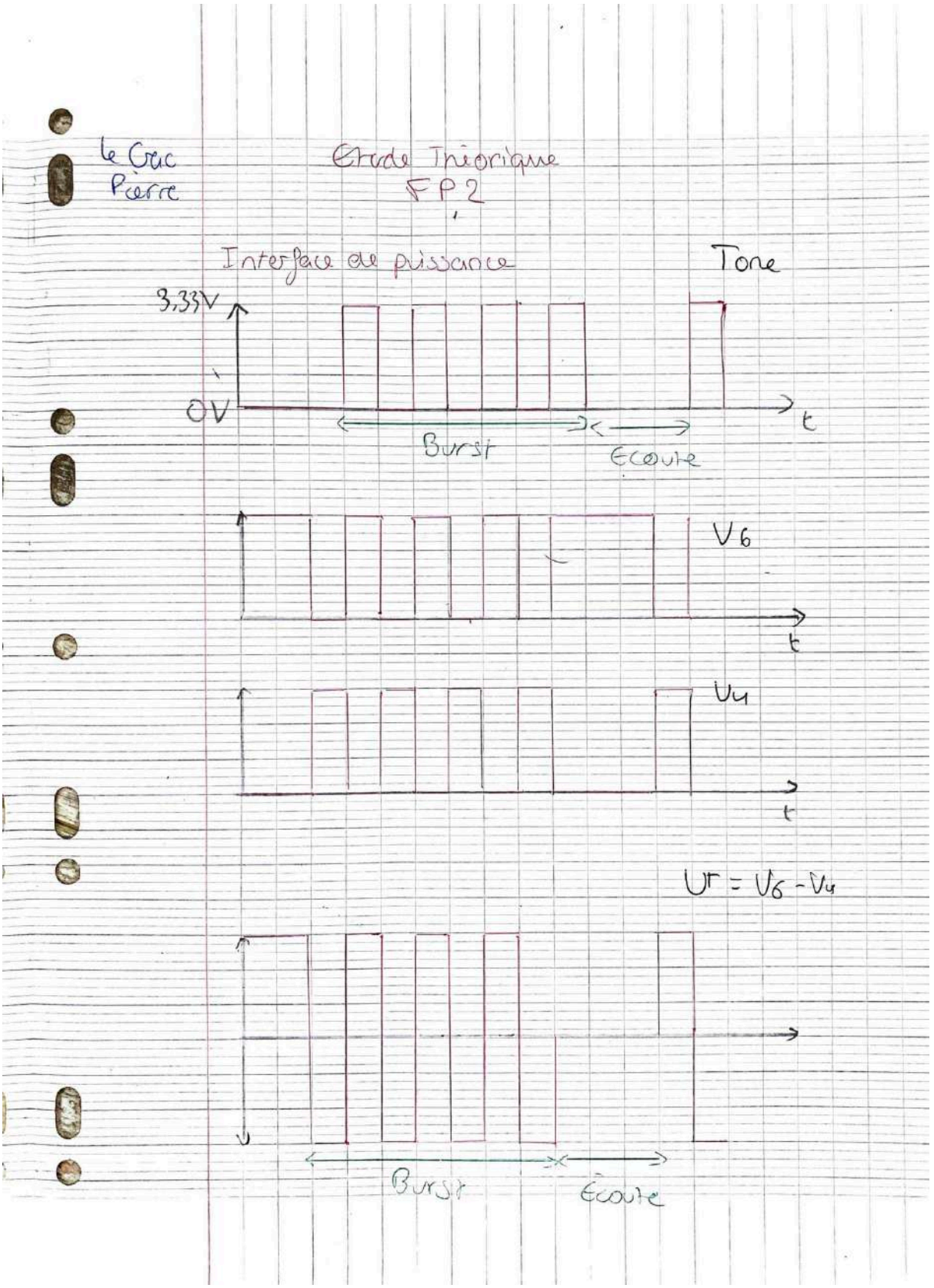
- Fonction FS2 : « Interface de puissance »

L'interface de puissance, composée de deux buffers inverseurs 74LVC2G14, permet de faire deux choses :

- Booster la sortance en courant du microcontrôleur SAMD21 initialement égale à 3mA et portée à 24mA afin de pouvoir piloter correctement le transducteur d'émission
- Doubler la dynamique de la tension aux bornes du transducteur d'émission afin d'augmenter la puissance d'émission de l'onde ultrasonore et ainsi augmenter la distance maximale mesurable par le sonar.







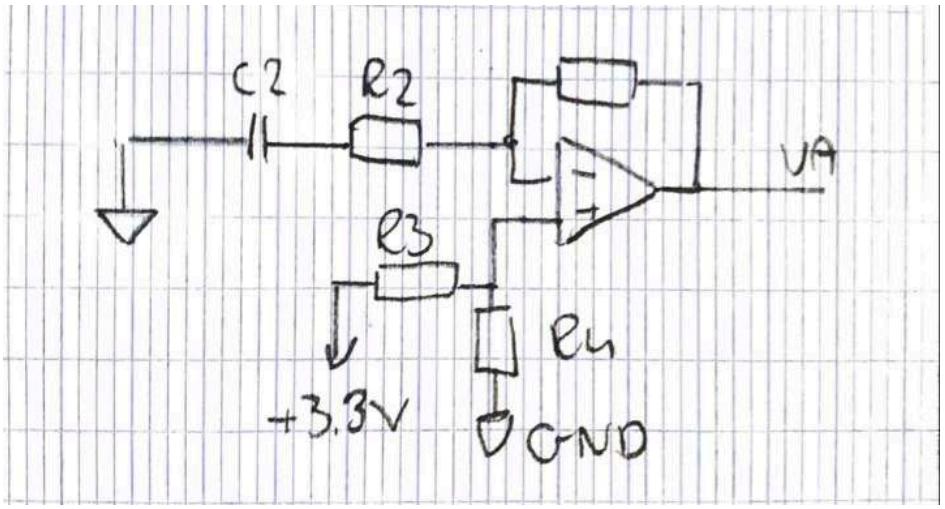
Vt=V6-V4 donc à un temps t où V6=3.3V et V4=0,Vt=3.3V
A un autre temps ou V6=0 et V4=3.3V, Vt=-3.3V.

- Fonction FS3 : « Amplification »

Le transducteur US de réception traduit le signal ultrasonore en une tension électrique dont le niveau est d'autant plus faible que l'obstacle est éloigné du sonar. Il faut donc amplifier cette tension afin de la rendre exploitable. On propose de mettre un œuvre un amplificateur de tension à AOP afin d'effectuer cette opération. On précise que celui-ci devra être alimenté en mono-tension 0/3.3V (unique tension disponible sur la carte).

Analyse statique

Pour cette analyse, on considérera que le transducteur de réception est inactif (aucune réception en cours) et qu'il est donc équivalent à une source de tension éteinte. On prendra, par contre, en compte l'influence de la source de tension de 3,3V appliquée sur la résistance R3.



Sur le pin 3 (borne +):

Avec Millman, $V\ += \frac{\frac{V}{R3}+\frac{0}{R4}}{\frac{2}{R3+R4}} = 1.65V$

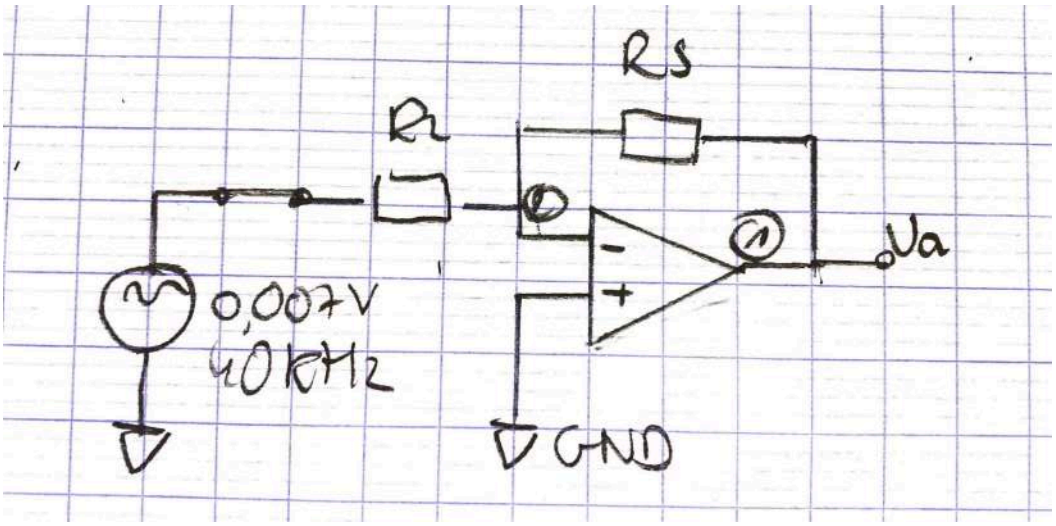
Sur le pin 2 (borne -);

Il y a pas de tension donc 0V

En sortie VA (pin 1) de l'amplificateur on a la valeur d'alimentation de l'AOP soit 3.3V. L'AOP est bien polarisé, la tension que nous voulons amplifier passe par la borne -. C'est donc un AOP amplificateur non inverseur.

Analyse dynamique

Pour cette analyse, on considérera que le transducteur de réception est actif (réception d'un signal US en cours) et qu'il est donc équivalent à une source de tension sinusoïdale d'amplitude 7mV et de fréquence 40kHz. La source de tension de 3,3V appliquée sur la résistance R3 est, dans ce cas, considérée comme étant éteinte.

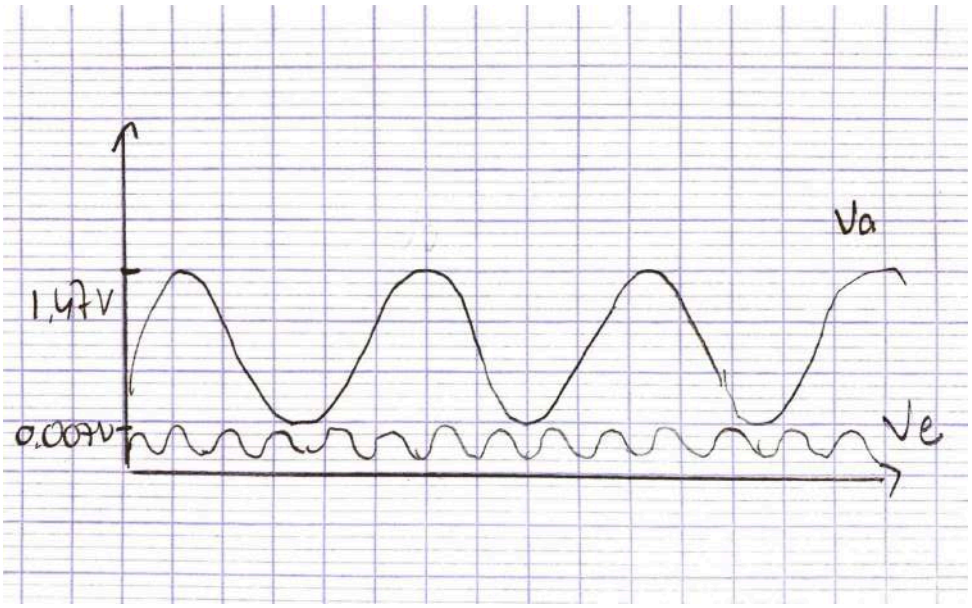


C2 = 1/jCw=1/(100n*2PI*40000)=39

R2 = 3900, on peut donc négliger C2

On a donc : Va=R5/R2*Ve=820k/3.9k*Ve
Le signal d'entrée est amplifié de 210. VAmax=210*0.007=1.47V

Va=1.47sin(2PI*40000*t)



Analyse fréquentielle

On rappelle que la transmittance de ce montage est de la forme suivante :

$$T = k.\frac{j\frac{f}{fcb}}{1+\frac{jf}{fcb}}.\frac{1}{1+\frac{jf}{fch}}$$

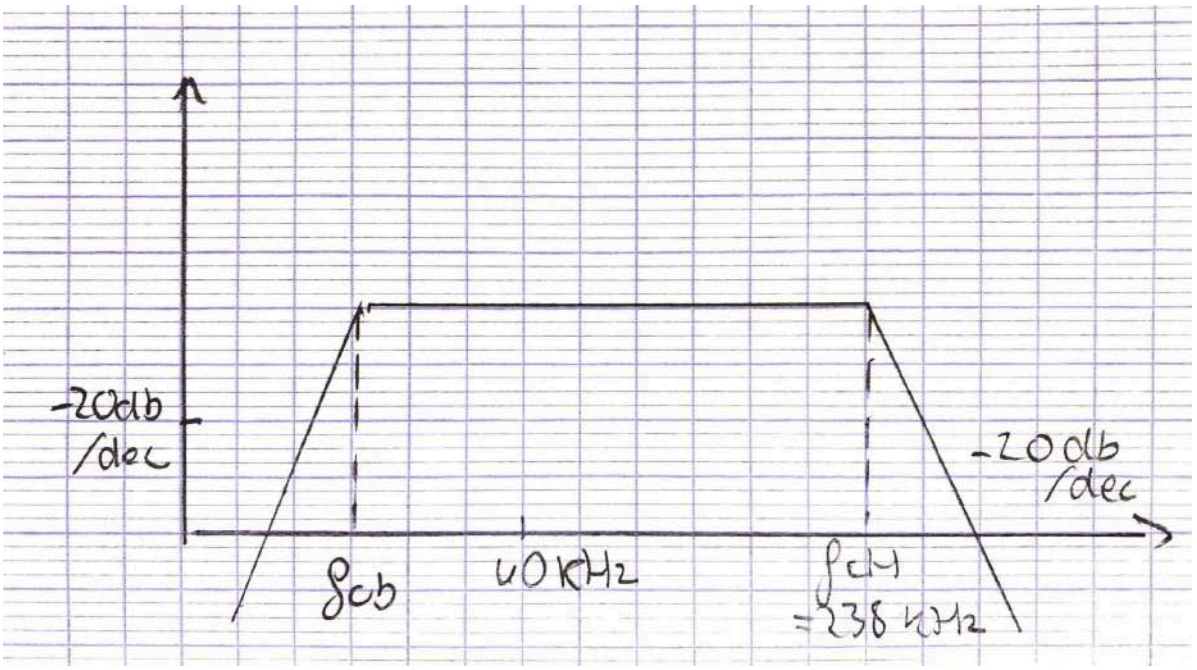
K =R5/R2=820000/3900=210

FCB =RCw=3900*100e-9*2Pi*40000=98Hz

FREQUENCY RESPONSE				
GBW	Gain-bandwidth product	V_S = 5 V	50	MHz

On a donc GBW=50Mhz

Fch=GBW/k=50Mhz/210=238khz



- Nous avons une bande passante de 98 Hz à 238 Khz. Notre signal de 40kHz est compris dedans. Ve sera donc bien amplifié.

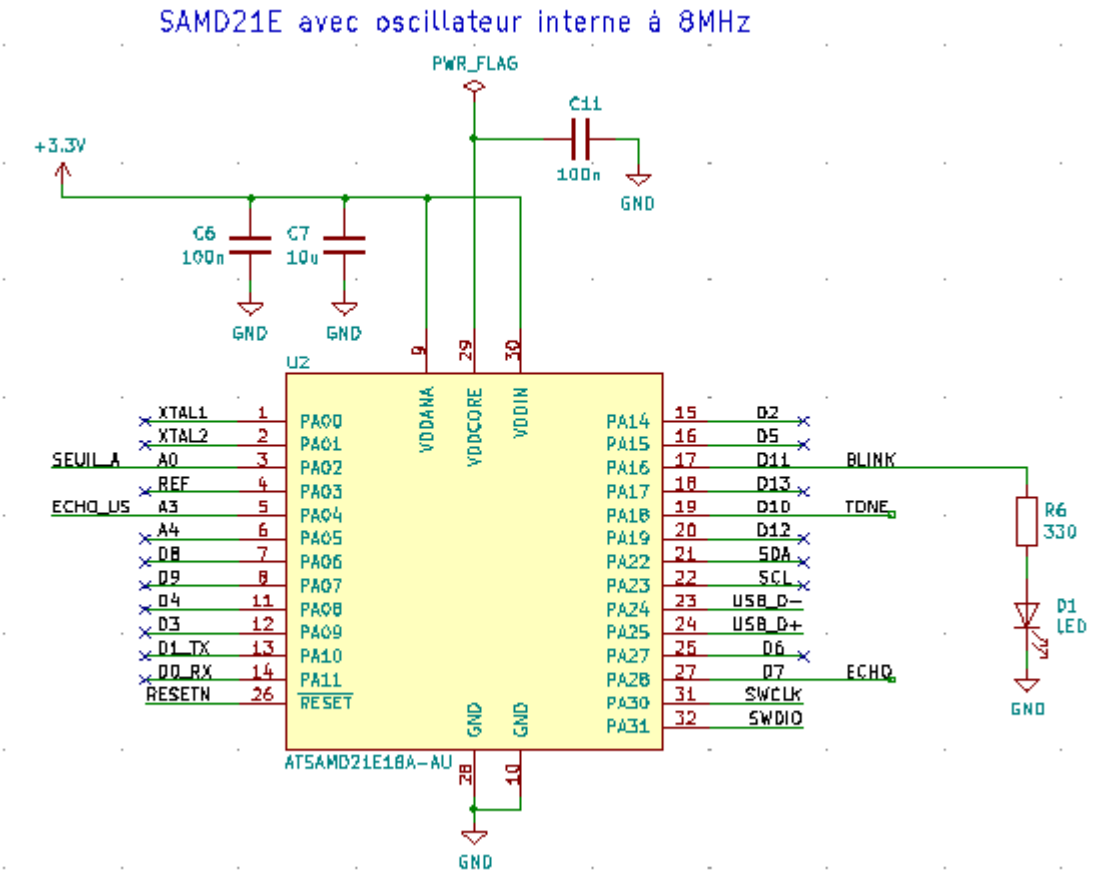
4- Conception du sonar

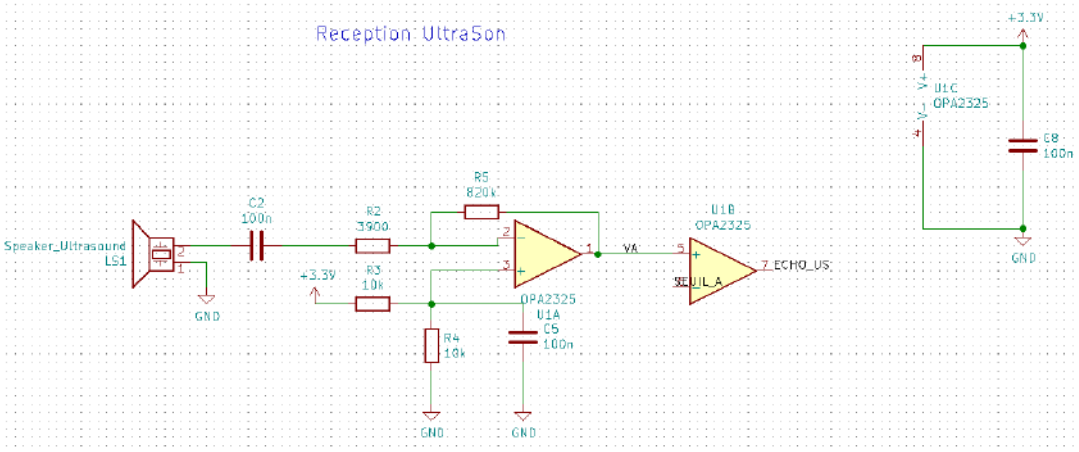
Pour commencer la conception du sonar, nous commençons par saisir le schéma structurel sous “eeschema” de kicad.

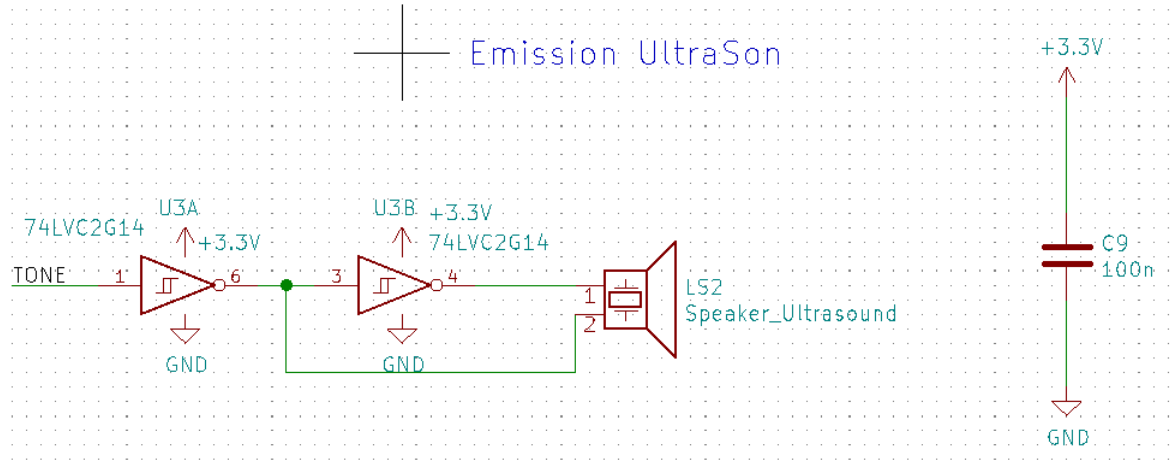
On précise que le sonar sera muni de deux connecteurs :
- Un connecteur “carte” 4 pins de type Groove (pas de 2mm) pour assurer la communication du sonar avec la fonction “gestion du robot”

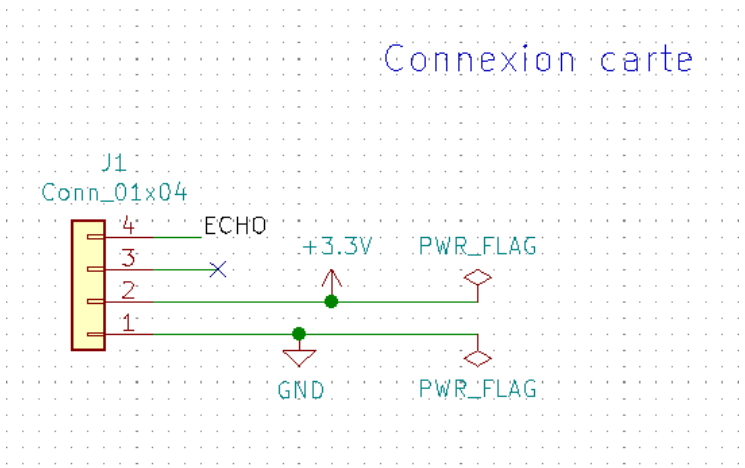
- Un connecteur “sonde” 2*3 pins (pas de 2,54mm) pour permettre la programmation du microcontrôleur.

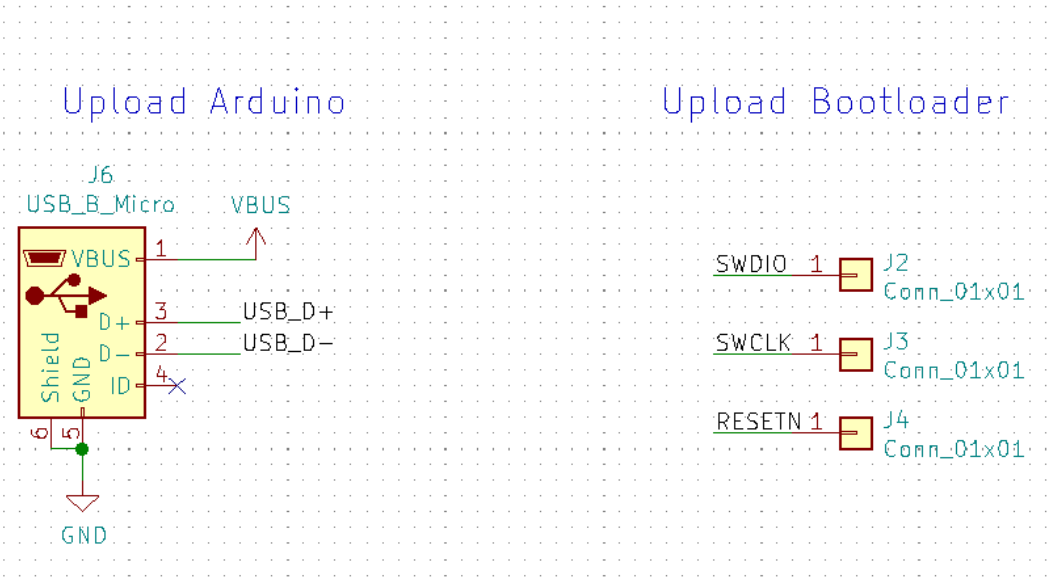
Une fois la saisie du schéma terminé, nous obtenons le résultat suivant :



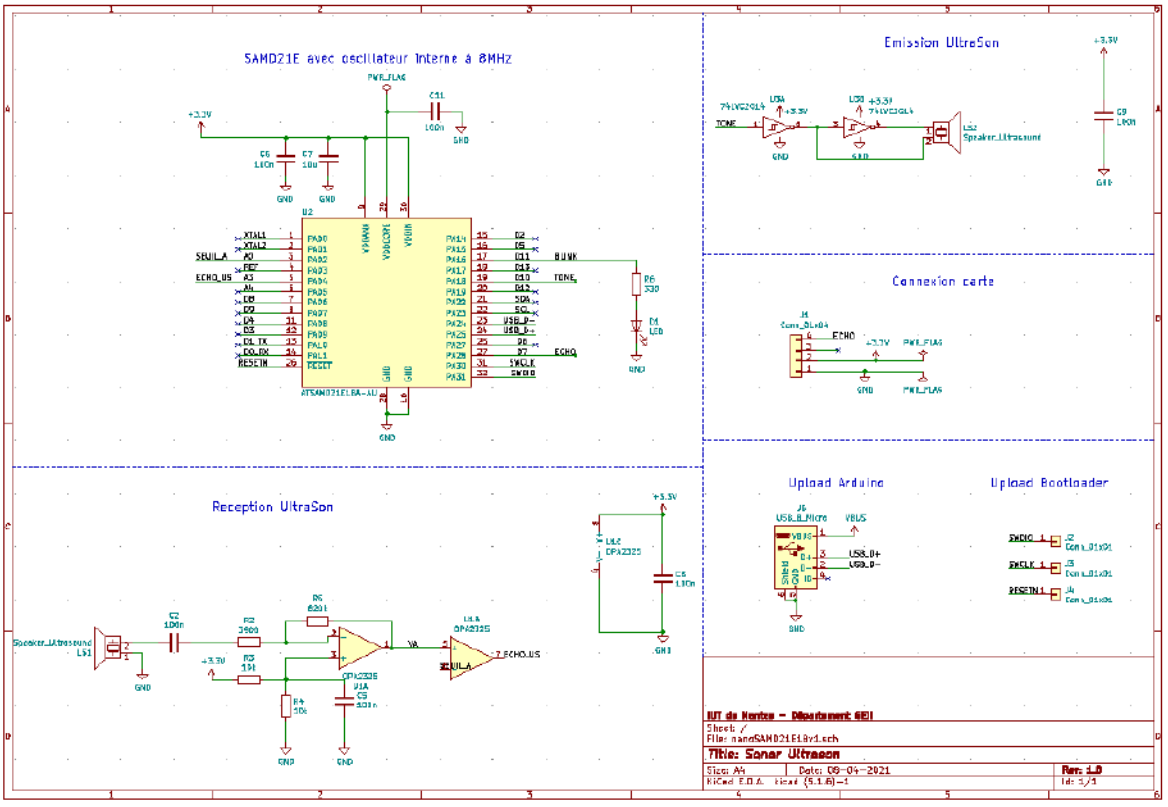

Reception UltraSon
Speaker_Ultrasound
LS1
C2
100n
R2
390Ω
R5
820k
R3
10k
R4
10k
+3.3V
GND
OPA2325
U1A
C5
100n
VA
U1B
OPA2325
SEUIL_A
7 ECHO_US
U1C
OPA2325
C8
100n


Emission UltraSon
U3A
74LVC2G14
+3.3V
TONE
1
6
GND
U3B
+3.3V
74LVC2G14
3
4
GND
LS2
Speaker_Ultrasound
1
2
+3.3V
C9
100n
GND


Connexion carte
J1
Conn_01x04
4
3
2
1
ECHO
+3.3V
PWR_FLAG
GND
PWR_FLAG



Voici le schéma général de ma carte:

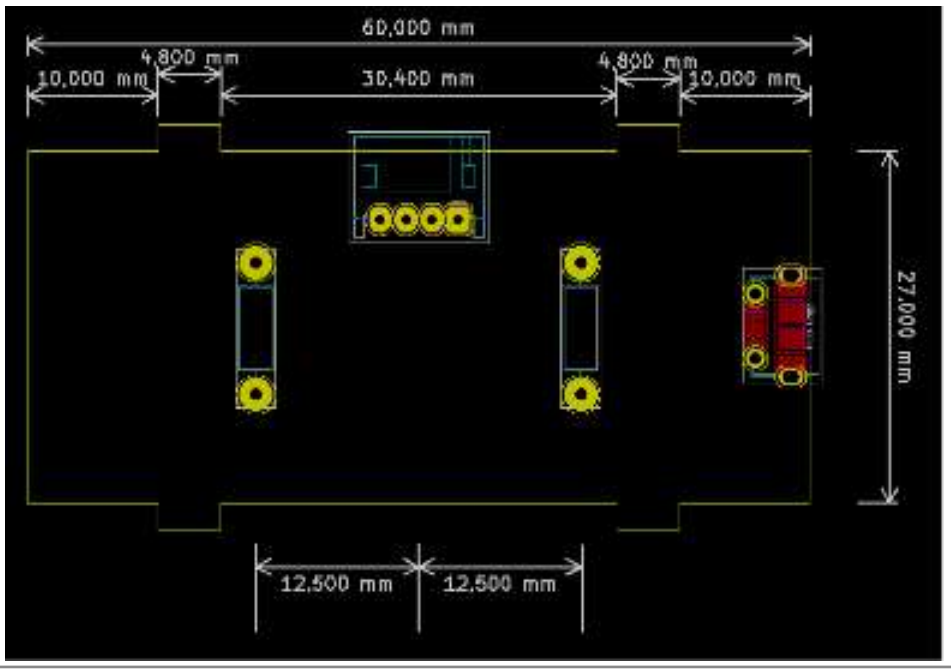


Après avoir effectué un contrôle des règles de conception, nous avons associé à chaque composant, une empreinte. Pour cela, nous avons dû nous référer au tableau suivant :

Composants	Librairie	Empreinte
Résistances	Resistor_SMD	R_0805_2012Metric
Condensateurs	Capacitor_SMD	C_0805_2012Metric
Led	LED_SMD	LED_0805_2012Metric
SAMD21E	Package_QFP	TQFP-32_7*7mm_P0.8mm
OPA2365	Package_SO	SOIC-8_3.9*4.9mm_P1.27mm
74LVC2G14	Package_TO_SOT_SMD	SOT-23-6
Connecteur USB	Connector_USB	USB_Micro-B_Amphenol_10103594-0001LF_Horizontal
Connecteur 1pin	Connector_PinHeader_1.00mm	PinHeader_1*01_P1.00mm_Vertical
Connecteur Carte	Connector_JST	JST_PH_S4B-PH-K_1x04_P2.00mm_Horizontal
Transducer	Resistor_THT	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal

Une fois l'association des empreintes terminée, nous avons généré la netlist puis ouvert PCBNEW afin de commencer le routage de la carte.

Durant cette étape, nous devions faire attention aux contraintes suivantes:



o Isolation : 0.25mm
o Largeur Pistes : 0.35mm
o Diamètre Via : 2mm
o Perçage Via : 0.6mm o Cliquer sur OK

• Importer les composants dans PCBnew et modifier les pads des composants suivants :

o Connecteur Groove 4 pins J1 : Pads de forme ovale
« Taille X » à 1,7mm, « Taille Y » à 2mm et « Taille du perçage » à 0,8mm

o Transducteurs LS1 et LS2 : Pads de forme circulaire
« Taille X » à 2,5mm et « Taille du perçage » à 1mm

o Connecteur 1 pin J2, J3, J4 : Pads de forme rectangulaire
« Taille X » à 2mm, « Taille Y » à 2mm et « Taille du perçage » à 0,6mm.

o Connecteur USB : Pads de forme ovale
« Taille X » à 1,5mm, « Taille Y » à 1,1mm et « Taille du perçage » à 0,6mm

• Tracer le contour de la carte en sélectionnant la couche « Edge.Cuts ».

• Placer le connecteur carte, le connecteur USB et les deux transducteurs en priorité

• Placer les trois circuits intégrés avec leur condensateur de découplage
o 74LVC2G14 au plus près du transducteur d'émission

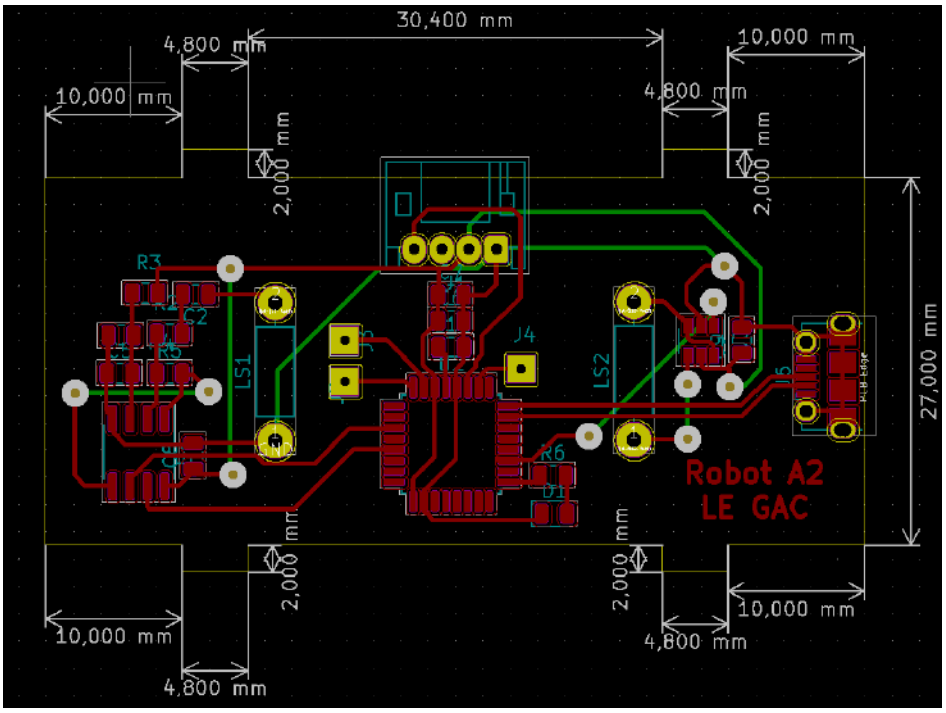
o SAMD21E entre les deux transducteurs

o OPA2365 au plus près du transducteur de réception

• Placer les résistances et condensateurs

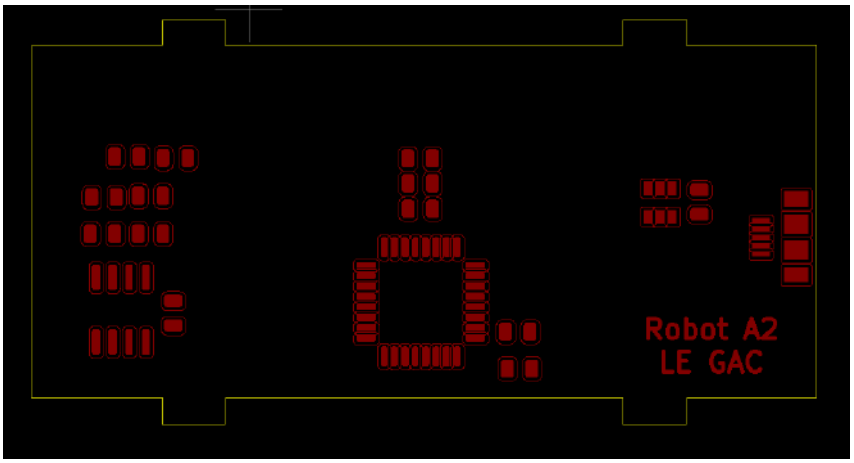
• Router les pistes (limiter le nombre de vias à 10)

Une fois le routage terminé, nous obtenons la carte suivante :

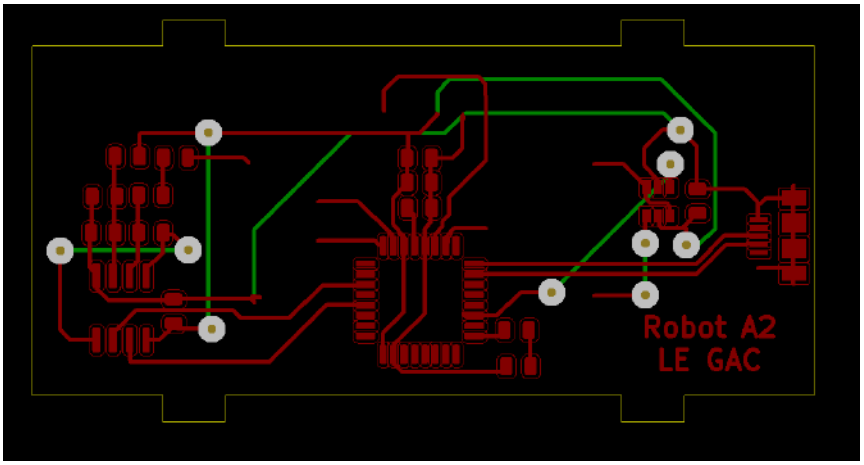


Voici les différentes couches :

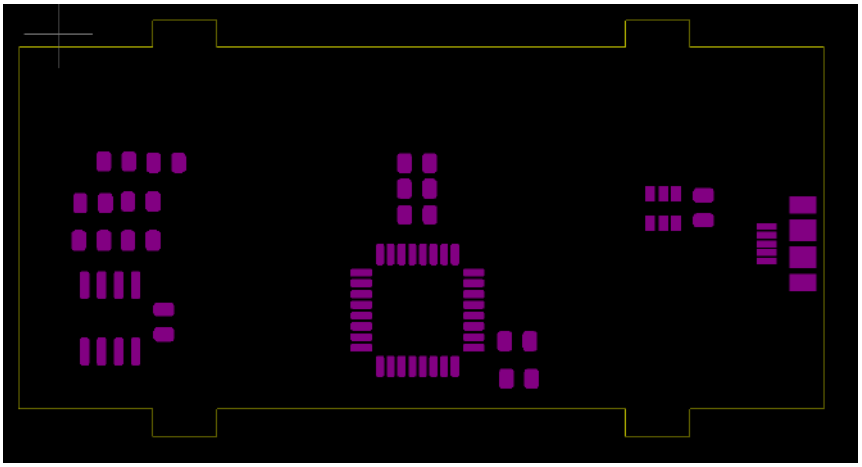
Pastilles:



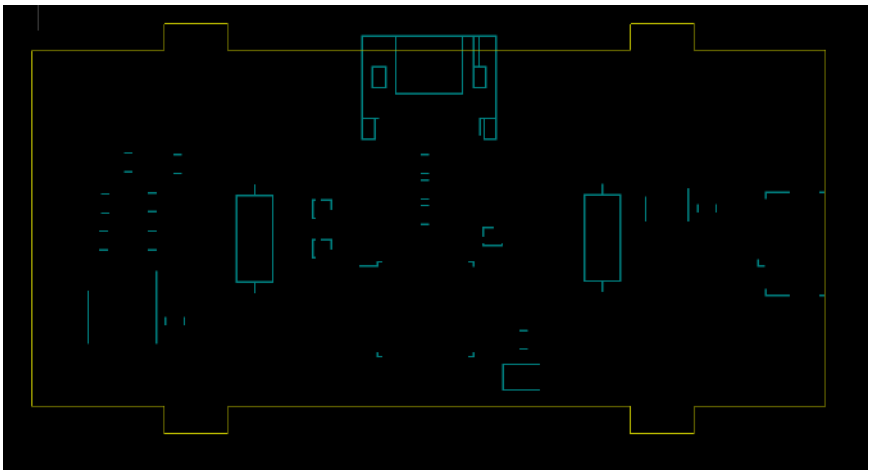
FcU et bCu:



FMask et BMask:

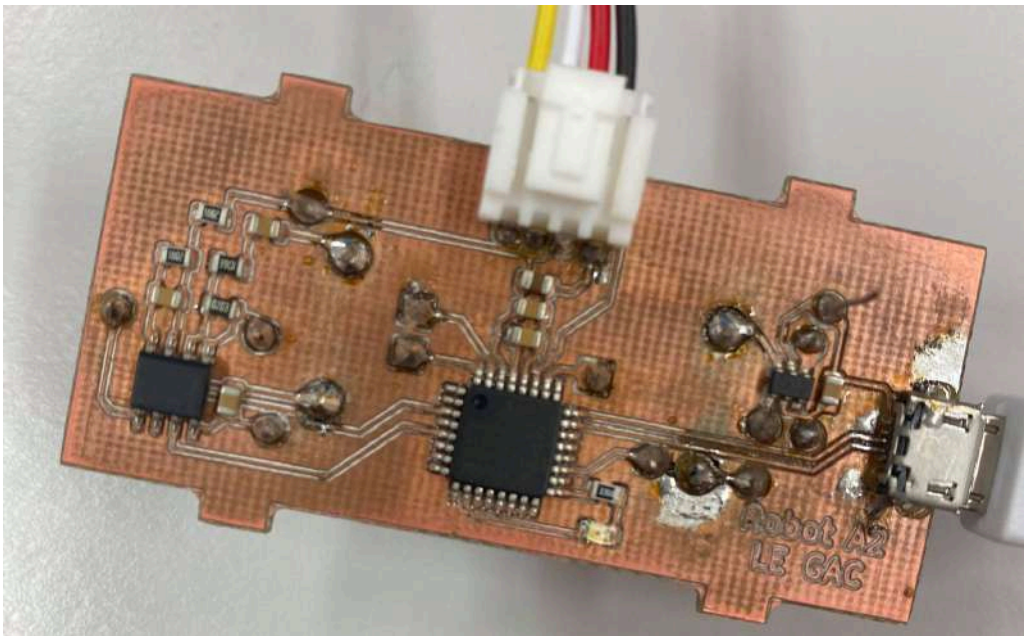


Couche FSilks:



5- Fabrication de la carte

Après avoir fourni au technicien les fichiers de mon projet (gravure de la carte) et après la réception de la carte, je dépose la pâte à braser aux endroits nécessaires.
Par la suite, je place les composants CMS et je place la carte au four.
Je réalise ensuite les vias et je soude les différents composants.
Voici ma carte finale:



6- Tests & Mesures

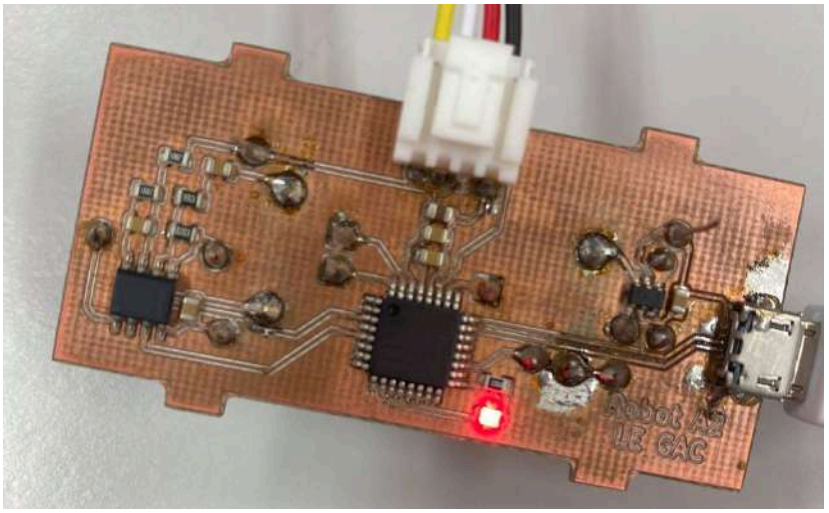
Ma carte est directement détectée mais elle est mal reconnue par l'ordinateur. Cela venait d'un problème du connecteur micro USB qui était mal soudé.

Après avoir corrigé les soudures et inspecté les pistes de ma carte, je la rebranche.

Cette fois-ci, elle est bien détectée. Je mets le BootLoader dans ma carte et j'insère le programme permettant de faire clignoter la LED.

```
void setup() {
 // initialize digital pin LED_BUILTIN as an output.
 pinMode(16, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
 digitalWrite(16, HIGH);   // turn the LED on (HIGH is the voltage level)
 delay(1000);                       // wait for a second
 digitalWrite(16, LOW);    // turn the LED off by making the voltage LOW
 delay(1000);                       // wait for a second
}
```

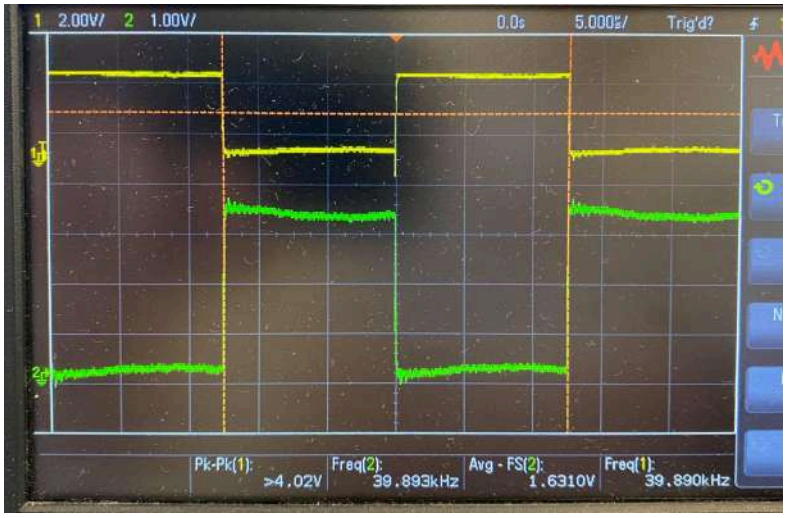
Ainsi j'obtiens:



Par la suite, j'émets un signal 40KHz sur l'émetteur de mon sonar grâce à la fonction suivante:

```
void setup()
{
 Serial.begin(9600);
 pinMode(18, OUTPUT);
}

void loop()
{
tone(18, 40000);
  delay(200);
}
```

Ainsi, quand je relève le signal à l'oscilloscope, j'obtiens un signal à 40KHz



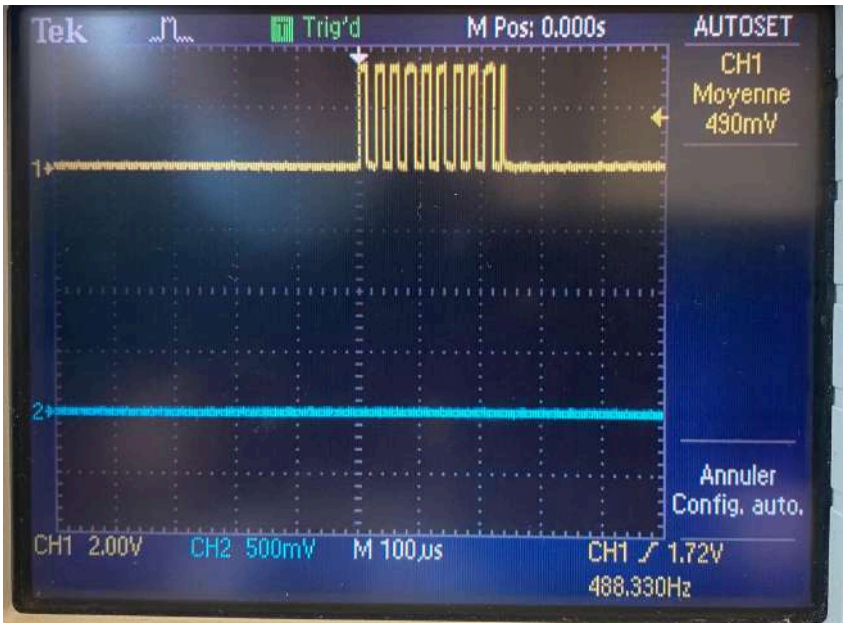
Cela prouve bien que ma carte fonctionne.

Par la suite, j'utilise ce code afin de voir si ma réception est bonne:

```
void setup()
{
 pinMode(18, OUTPUT);
 pinMode(4, INPUT);
 pinMode(2, INPUT);
}

void loop()
{
tone(18, 40000);
  delayMicroseconds(200);
  noTone(18);
  delay(20);
  analogWrite(2, 210);
}
```

Ce programme va permettre de visualiser Va sur l'oscilloscope qui va alors se modifier en fonction de la distance des sonars par rapport à un objet.

Ainsi nous obtenons les chronographes suivants:

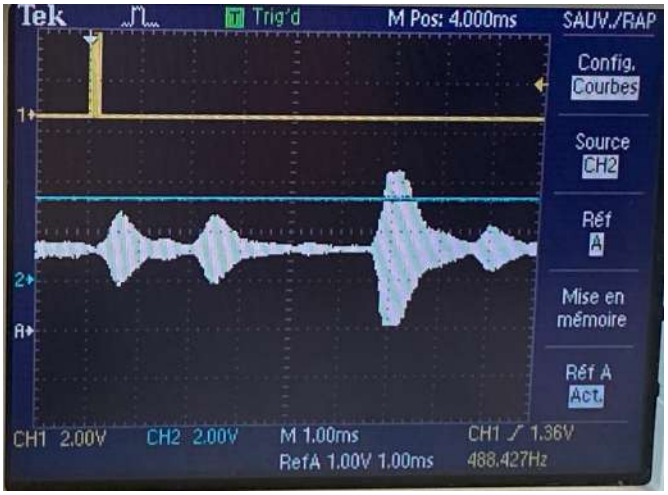


Nous pouvons remarquer que mon émission fonctionne (en jaune).
Ma réception quant à elle, ne fonctionne pas (en bleu).

Ce problème vient de la résistance R5 qui était reliée à la masse.
A cause de cela, l'amplificateur assurait un rôle de comparateur au lieu d'amplificateur.

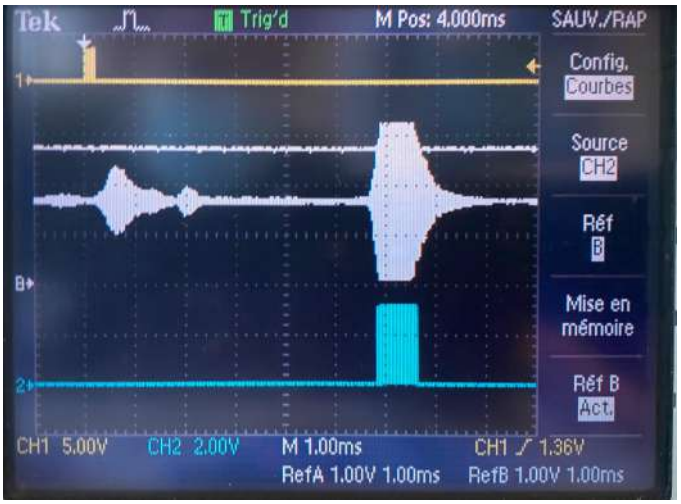
De plus, la pin 3 était à 0V au lieu de 1,6V, et la pin de sortie était à 3,3V au lieu de 1,6V, ce qui prouve un problème lié à la résistance R5.

Une fois le problème corrigé, nous obtenons:



Nous pouvons remarquer que le signal fonctionne. Comme nous pouvons le remarquer, il y a des rebonds. Nous allons utiliser de la mousse pour les enlever pendant nos tests.

Le signal bleu représente le seuil, quand le signal a dépassé ce dernier, le signal sera alors détecté.



La courbe jaune représente l'émission de mon signal.
Le signal continu blanc représente le seuil.

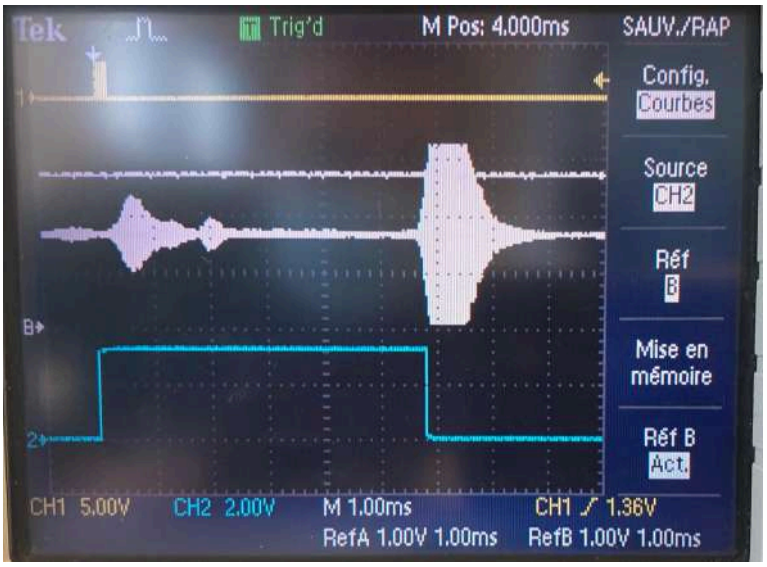
La courbe blanche représente la réception sur Va, nous pouvons remarquer que les rebonds ont diminué grâce à la mousse.
Notre signal bleu représente le signal Echo_US.

La réception et le seuil vont alors être comparés et lorsque la réception dépasse le seuil, le signal Echo_US va passer à 1.

L'ensemble des tests de ma carte montrent que cette dernière fonctionne.
Il ne reste plus que le code final permettant d'assurer le rôle de ma carte.
Pour cela nous allons mesurer la distance entre l'émission et le moment où Va dépasse le seuil.

Nous utilisons le code suivant:

```
#define TONE 18
#define ECHO 28
#define ECHO_US 4
#define VA 2
#define duree

void setup() {
 // put your setup code here, to run once:
 pinMode(TONE,OUTPUT);
 pinMode(ECHO_US,INPUT);
 pinMode(VA,INPUT);
 pinMode(ECHO,OUTPUT);
 analogWrite(VA,200);
 //pinMode(
}

void loop() {
 // put your main code here, to run repeatedly:
 digitalWrite(ECHO,HIGH);
 tone(TONE,40500);
 delayMicroseconds(200);
 noTone(TONE);
 while(digitalRead(ECHO_US)==0);
 digitalWrite(ECHO,LOW);
 delay(15);

//Serial.println(pulseIn(ECHO, HIGH));
//Serial.println(digitalRead(4));
}
```



Nous nous plaçons sur la PIN 28 du microcontrôleur, ainsi nous obtenons le signal en bleu.

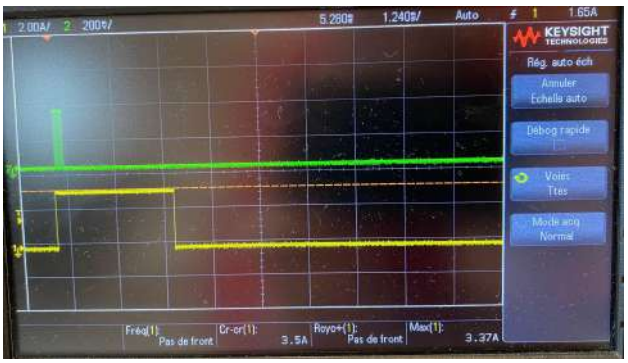
Nous remarquons que le signal commence en même temps que l'émission (en jaune) , et que ce dernier se termine lorsque la réception du signal dépasse le seuil.

Cette durée correspond à la durée à laquelle le signal du sonar va de l'émission à la réception, et dépend donc de la distance de l'objet par rapport au sonar.

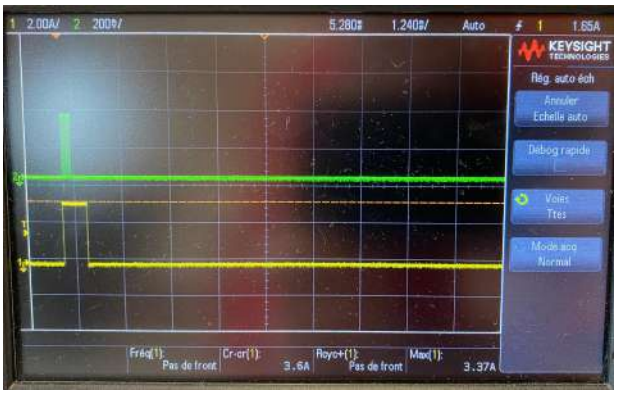
Code final:

```
#define TONE 18
#define ECHO 28
#define ECHO_US 4
#define LED1 16
#define VA 2
#define TIMEOUT 3000

unsigned T_Init;
void setup() {
 // put your setup code here, to run once:
pinMode(TONE,OUTPUT);
pinMode(ECHO_US,INPUT);
pinMode(VA,INPUT);
pinMode(ECHO,OUTPUT);
analogWrite(VA, 200);
//Serial.begin(9600);
//pinMode(
}


// the loop function runs over and over again forever
void loop() {
//  digitalWrite(ECHO,HIGH);
 T_Init= micros();
 digitalWrite(ECHO,HIGH);
 tone(TONE,40500);
 delayMicroseconds(200);
 noTone(TONE);
 while(digitalRead(ECHO_US)==0 && ((micros()-T_Init)<TIMEOUT));
 digitalWrite(ECHO,LOW);


 //while(digitalRead(ECHO_US)==0);
//  digitalWrite(ECHO,LOW);
 delay(20);
}
```

Voici ce que j'obtiens:

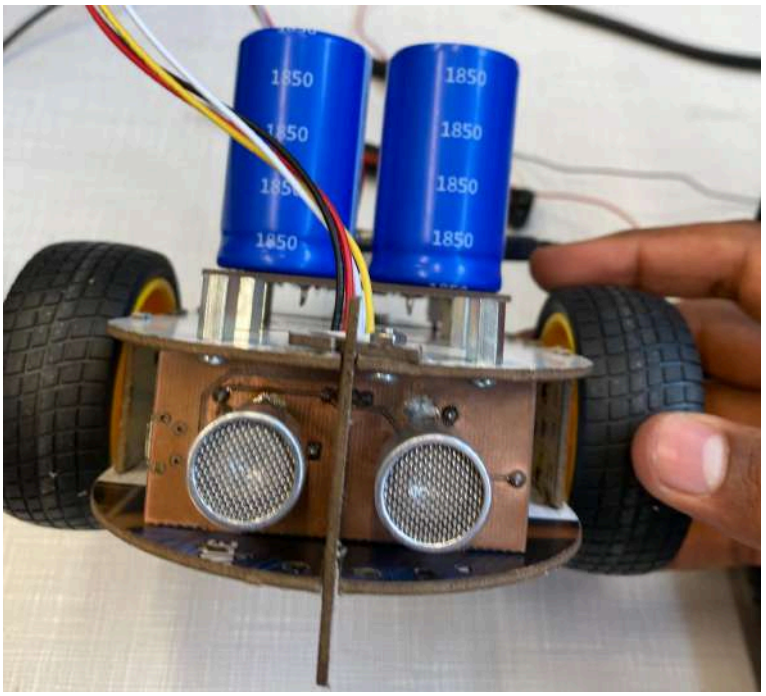


A une distance de 1m, le signal Écho à une grande période.



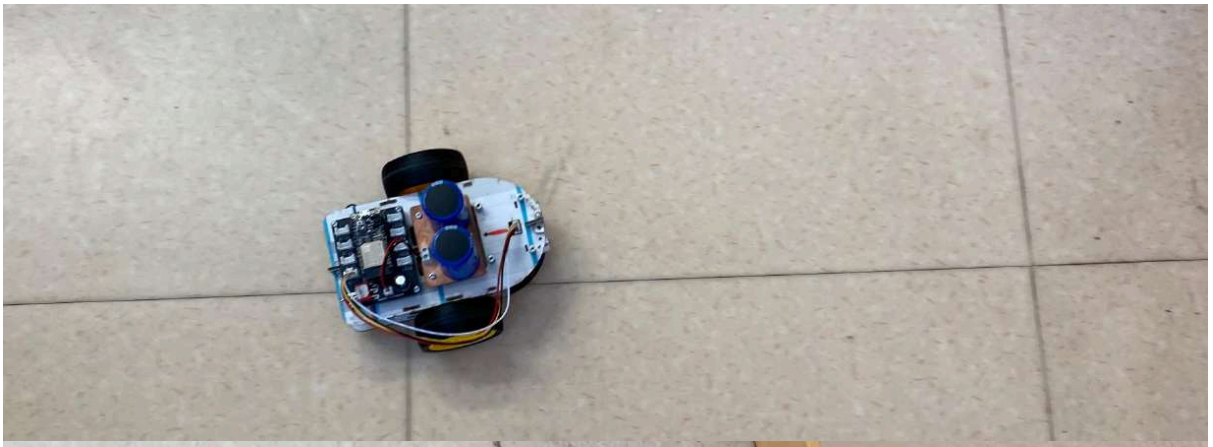
A une distance de 15cm , le signal Écho à une plus petite période.

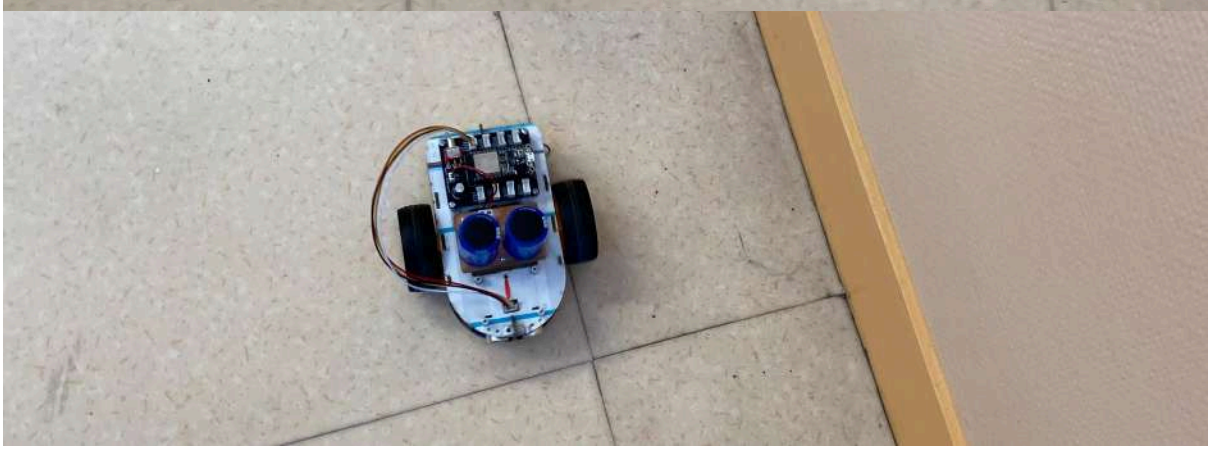
Mon programme fonctionne correctement.
Je vais ainsi pouvoir l'insérer sur le robot Léo.



Une fois mon sonar implémenté sur ma carte, nous allons pouvoir tester si ce dernier fonctionne.

Après avoir paramétré nos fonctions FP1 et FP2 et adapté les programmes, nous testons notre robot.



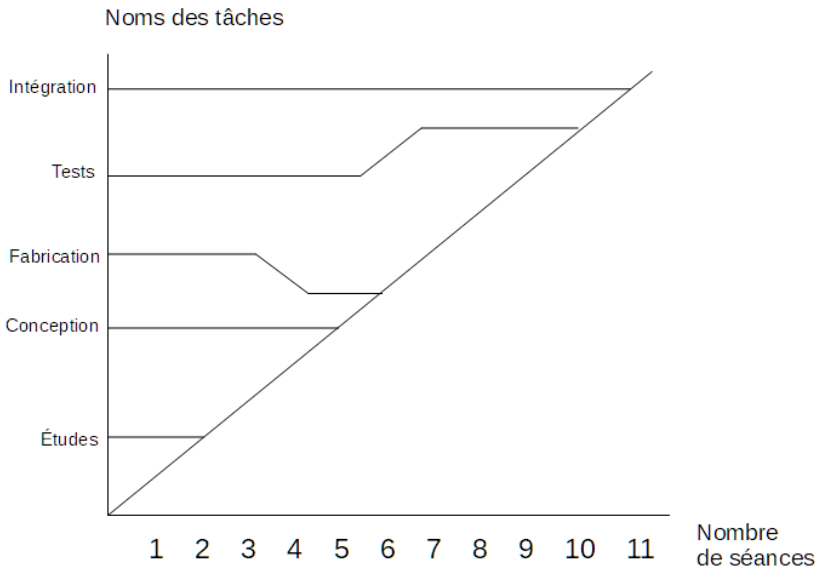


Nous pouvons remarquer qu'à une distance de 25cm, le robot change de direction et tourne instantanément à droite.

Cela prouve que ma fonction FP1 fonctionne bien.

J'ai donc bien réussi à faire fonctionner mon sonar.

Emploi du temps:



Voici mon avancement réel du projet.

Nous pouvons remarquer que j'ai fait mes études, ma conception et mon intégration avec le temps adapté.

Cependant, j'ai pris 1 séance d'avance sur ma fabrication, mais une séance de retard sur mes tests.

Cela n'a pas impacté sur la fin du projet qui s'est quand même terminé séance 11.

Conclusion de la fonction FP1:

J'ai beaucoup apprécié ce projet. Il m'a permis de mélanger aussi bien la théorie et la pratique. La phase de conception était très intéressante et j'ai préféré cette dernière à la partie Arduino.

Travailler en groupe est quelque chose que j'aime beaucoup et c'était un honneur de partager un grand projet avec 3 membres de ma classe.

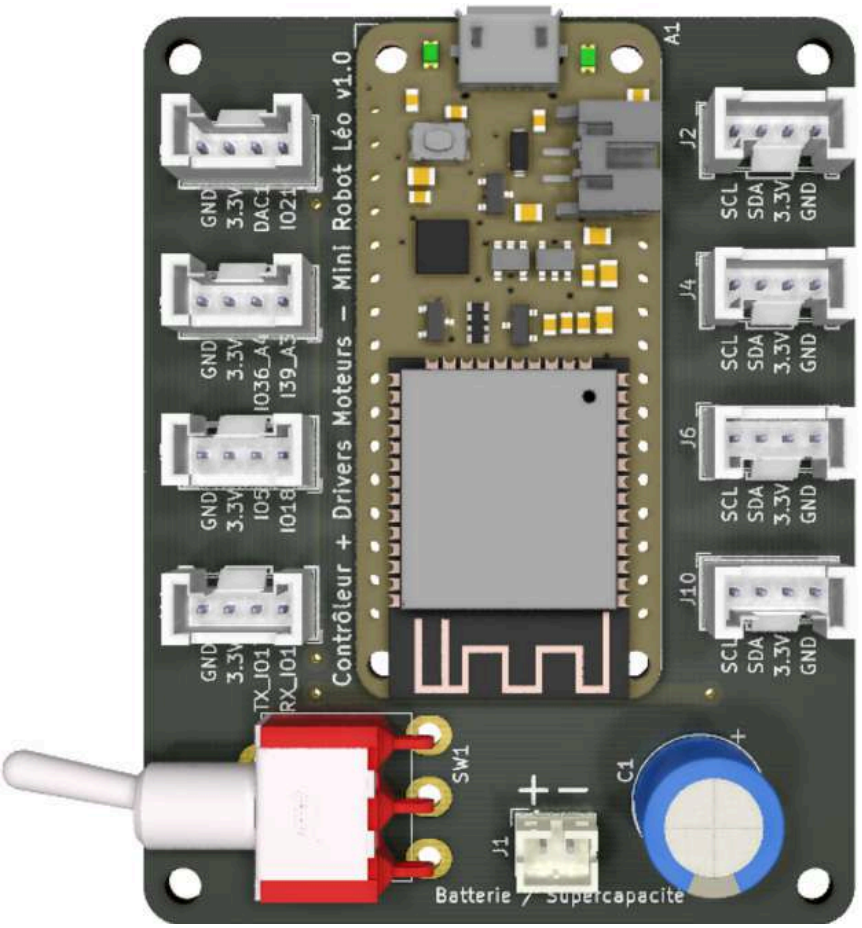
Je n'ai pas rencontré de difficulté particulière pendant la conception de ma carte. Avec de la concentration et de la réflexion, j'ai rapidement trouvé d'où venait les erreurs et j'ai tout mis en œuvre pour qu'elle fonctionne.

L'essentiel de mes problèmes était durant la phase de programmation où mes programmes ne fonctionnaient souvent pas, et j'ai dû être persévérant afin de détecter les erreurs de mes codes, ce qui m'a permis d'améliorer mes compétences en informatique.

Voir notre robot éviter les obstacles était une grande fierté pour moi. J'ai donc réussi ma fonction FP1.

La fonction FP2 « Gestion du mini robot Léo »

(Théodros Maru)

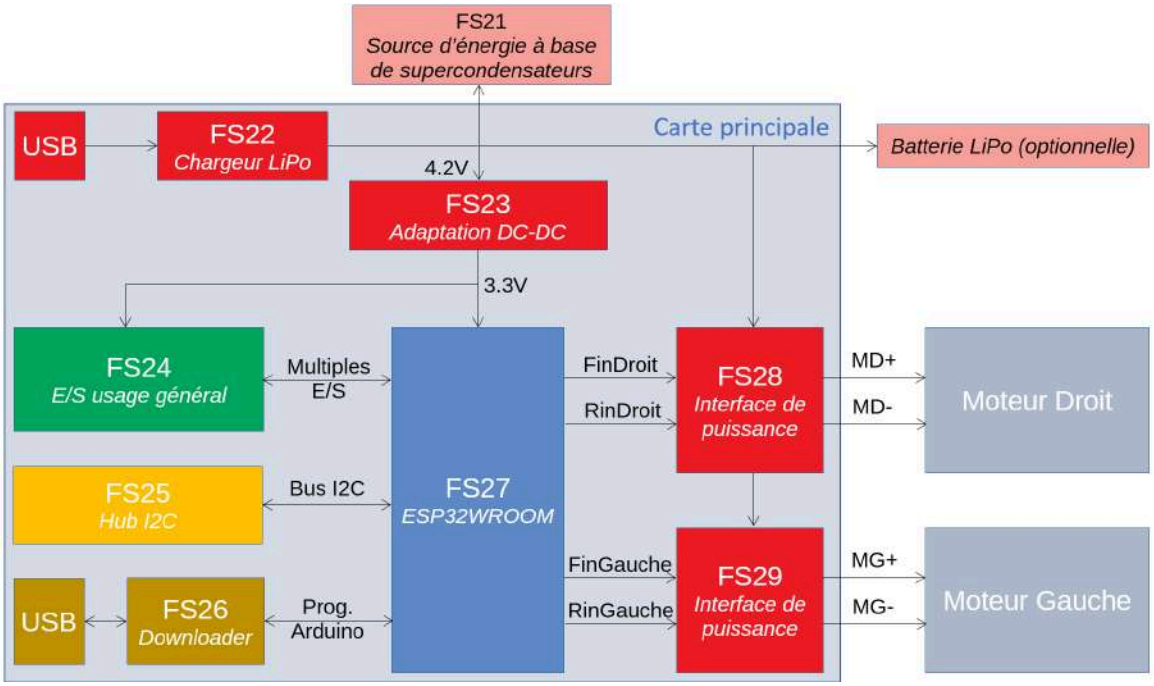


Le mini robot Léo repose sur une structure multi-cartes rattachées à une carte principale exploitant un ESP32 de la société chinoise Espressif.

Cette carte principale assure le traitement des informations envoyées par les différentes cartes qui y sont branchées afin de faire avancer le robot Léo vers la balise tout en évitant les obstacles. Elle s'occupe également de la régulation et de la distribution de l'alimentation de l'ensemble des éléments du robot.

La source d'énergie du robot viendra d'une carte secondaire à base de supercondensateurs capable de remplacer la batterie LiPo et dont le développement fait également partie du projet.

On peut résumer cela à la structure fonctionnelle suivante :



1 Questions préliminaires

1.1.1)

1)

Loi de Coulomb : « L'intensité de la force électrostatique entre deux charges électriques est proportionnelle au produit des deux charges et est inversement proportionnelle au carré de la distance entre les deux charges. La force est portée par la droite passant par les deux charges. » $F = (k * q_1 * q_2)/d^2$

2)

La loi liant la charge stockée sur l'électrode d'une capacité et la différence de potentiel à ses bornes :

Pour un condensateur donné, la charge Q portée par ses armatures est proportionnelle à la différence de potentiel V qu'on y applique.

La constante de proportionnalité de cette relation, C, est appelée capacité du condensateur :

Q = C*U (C = constante) . De plus $V = Q/(\varepsilon_0 * A)$

3)

L'expression permettant de calculer la durée notée ΔT nécessaire pour que la différence de potentiel aux bornes d'une capacité change de ΔV lorsque le courant I est constant est:

$$i = \frac{\mathrm{d}Q_1}{\mathrm{d}t} = C\frac{\mathrm{d}u}{\mathrm{d}t}.$$

ΔV = I.ΔT/Ceff

donc ΔT=(ΔV*Ceff)/I

4)

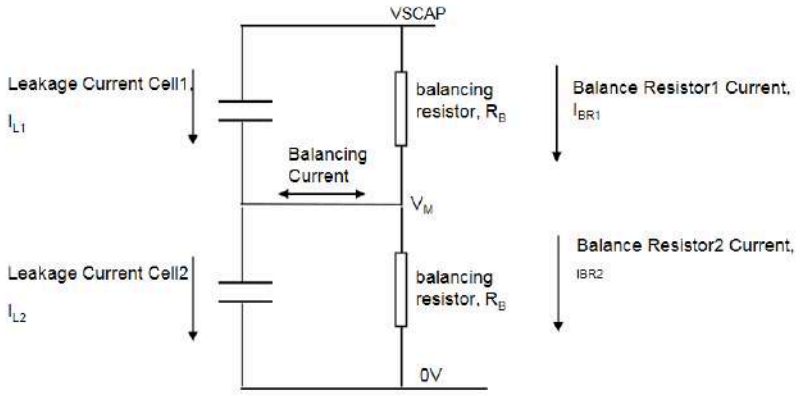
Application numérique :

ΔT pour I = 500mA , C = 200F lorsque V passe de 4.2V à 3.3V vaut :

ΔT = 200*0.9/0.5 = 360s

1.2.2 Etude

1) Après consultation des documents sur les super capacités nous en savons un peu plus et pouvons répondre aux interrogations suivantes:



- The purpose of this circuit is to maintain V_M close to $V_{SCAP}/2$.
- $V_M = R_B x I_{BR2} = R_B x (I_{BR1}$-Balancing Current).
- For this circuit to work, Balancing Current must be << I_{BR1}, I_{BR2}.
- VM must be prevented from going >> $V_{SCAP}/2$ or <<V_{SCAP} for any significant length of time.
- SIMPLE but HIGH CURRENT SOLUTION (~100µA)

a)

Il faut mettre en série les supercondensateurs pour obtenir un supercondensateur dont la tenue en tension soit le double des supercondensateurs utilisés.

b)

Les 2 supercondensateurs sont en série, donc le nouveau supercondensateur obtenu est de C = 200*200/(200+200)= 100F

2)

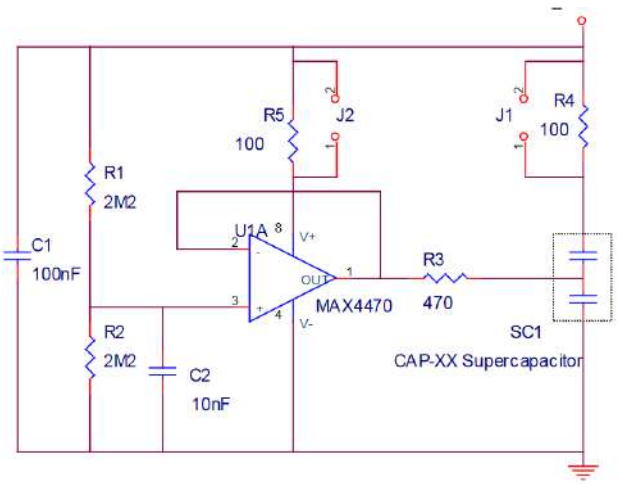
a)

Les 2 supercondensateurs placés en séries ont besoin d'un balancement de la tension, sinon une légère différence apparaîtra au niveau de la fuite de courant. Ceci peut conduire à un mauvais balancement de la tension, conduisant à une saturation en tension d'un des condensateurs.

b)

Les deux solutions pour régler ce problème sont:

- De mettre une résistance en parallèle de chaque cellule de condensateur afin d'éviter les fuites de courant.
- La deuxième solution est de créer un circuit de balancement comme ci-dessous

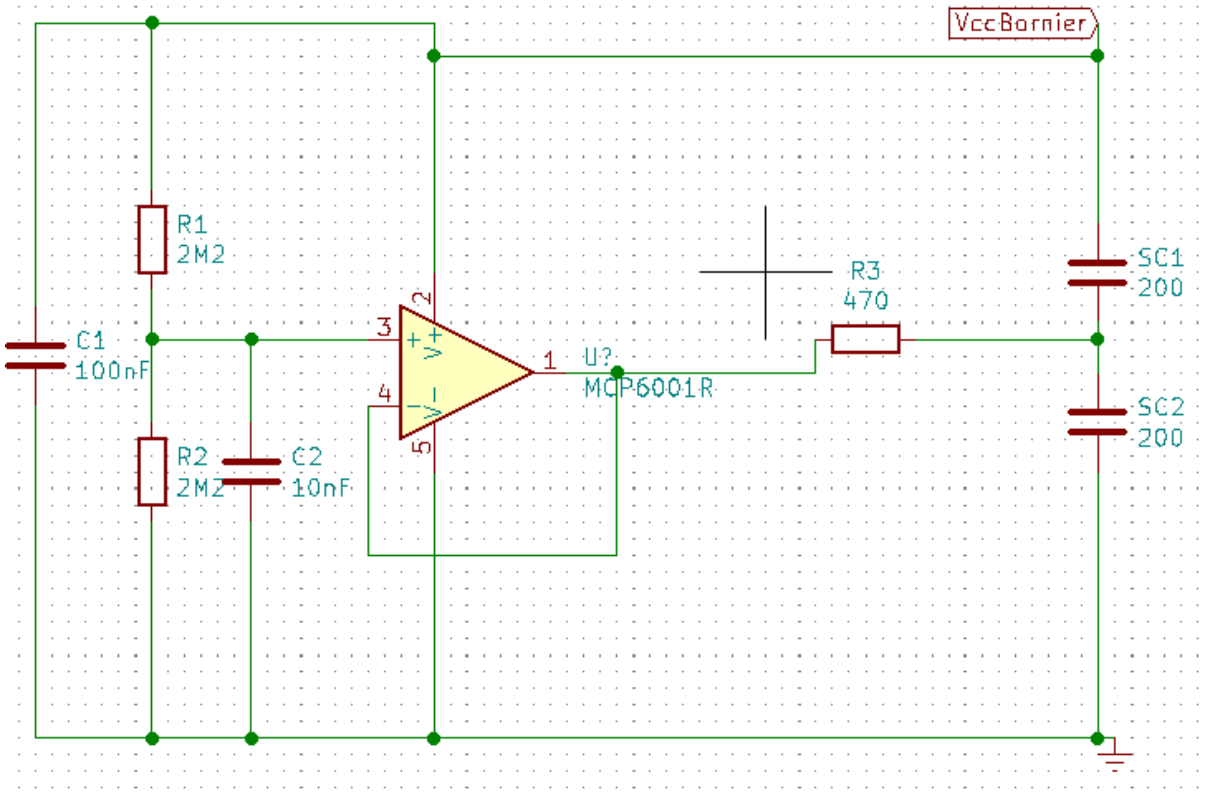


c)

1ere solution	2eme solution
Avantages - Simple Désavantages - Courant important	Avantages - faible consommation - balance plus facilement le courant entre les condo Désavantages - beaucoups de composants différents

On retient la solution la plus sophistiquée pour assurer le maintien de l'équilibre de charge des supercondensateurs.

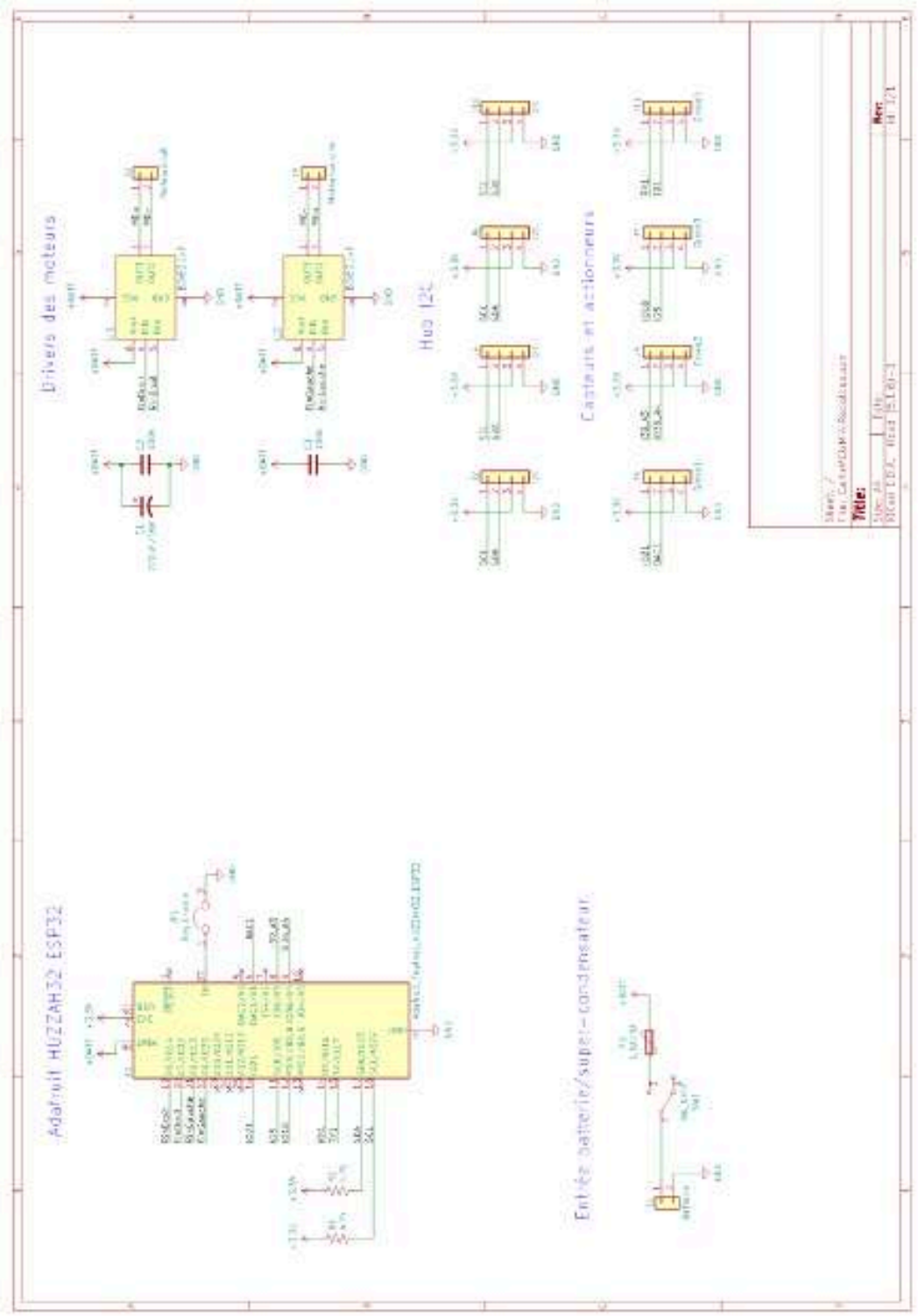
Donc la solution 2 sera retenue:



2 La carte principale

2.1Le schéma

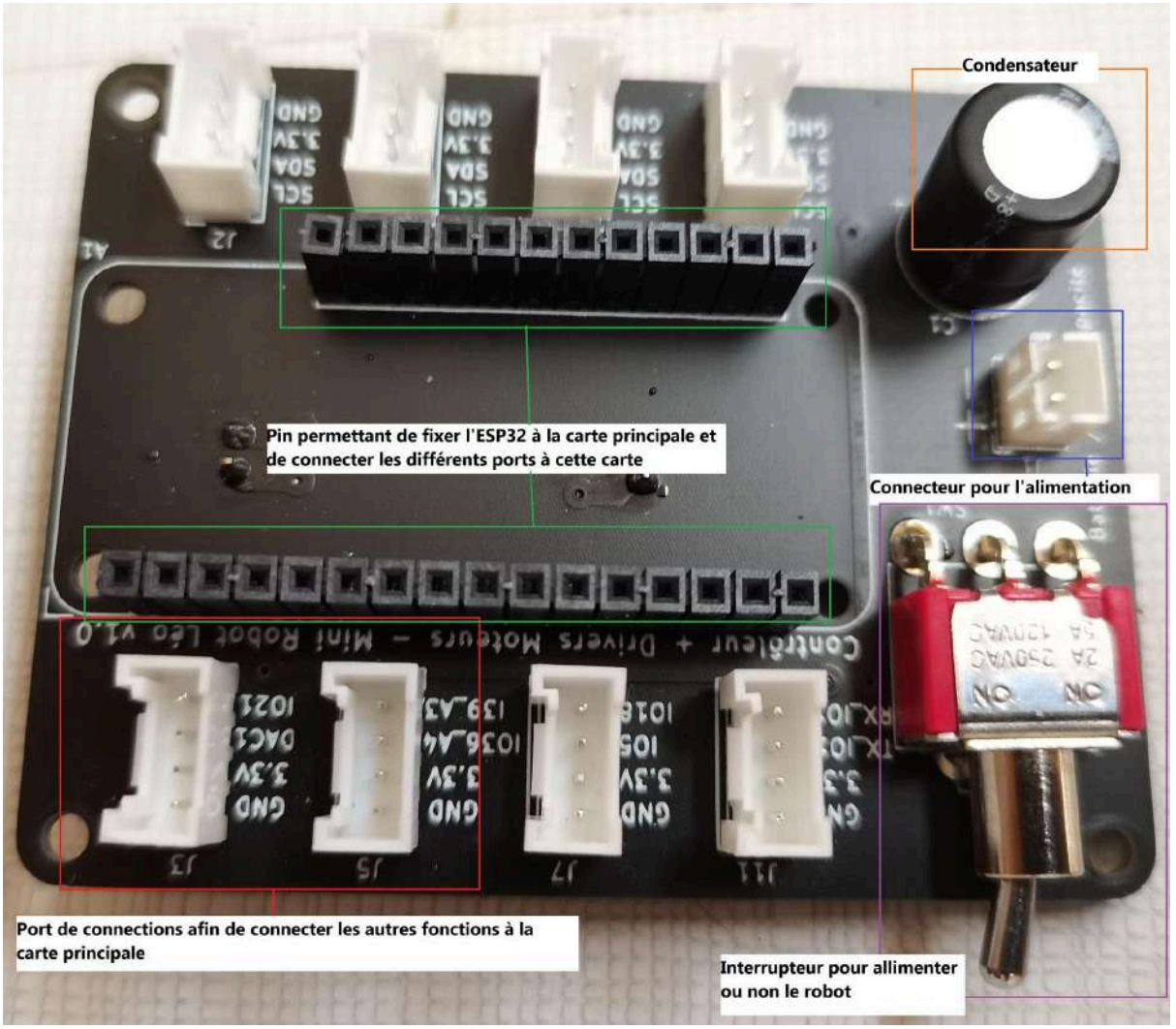
Le schéma de la carte est le suivant :



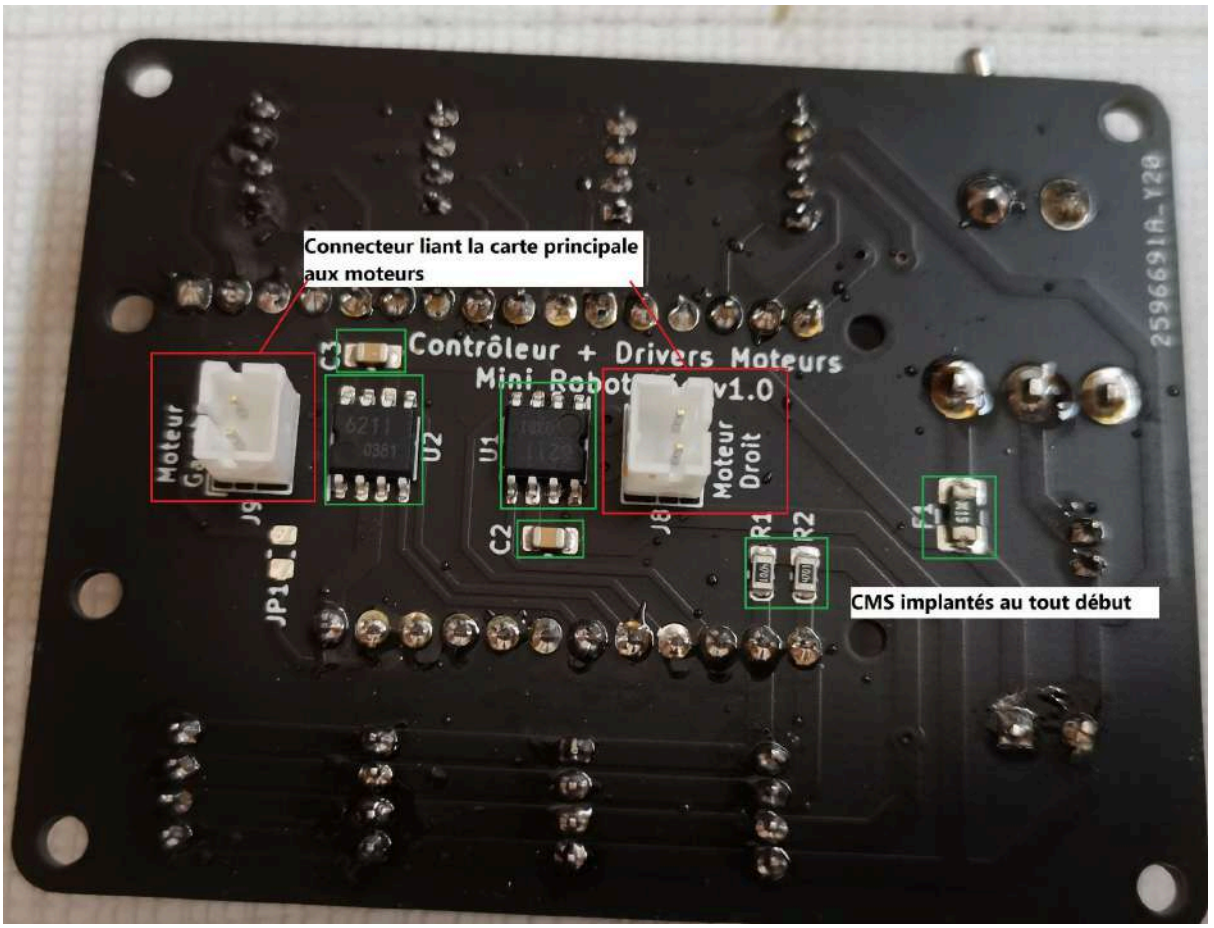
2.2Implantation des composants sur la carte

L'implantation des composants se fait sur les deux faces du circuit imprimé :

Dessus de la carte:



Dessous de la carte:



Démarche:

- Nous avons d'abord mis de la pâte à braser sur l'emplacement des CMS afin d'ensuite les mettre au four pour les souder.
- Nous avons ensuite soudé tous les composants traversants afin de contrôler les différentes parties du robot.

3 Structure du Robot

Pour faire la structure du robot nous avons utilisé des calendriers en cartons que l'IUT n'utilisait pas afin de recycler les calendriers qui étaient normalement jetés tous les ans. Nous avons donc utilisé une machine à découpe laser afin de découper la forme du robot voulu dans le carton.

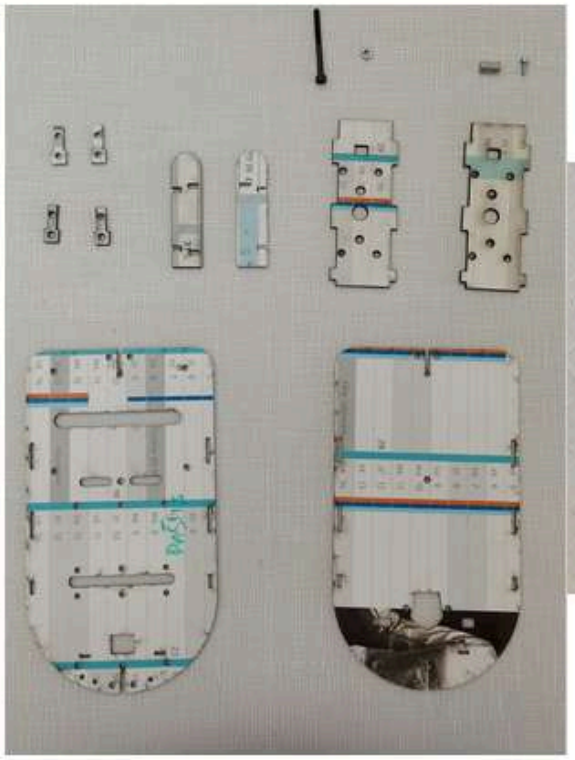
Le procédé était donc le suivant :

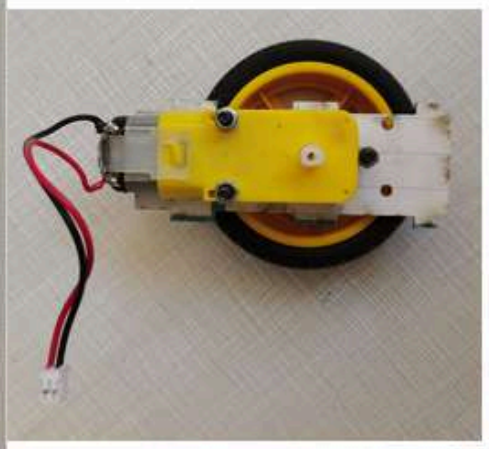
- Dans un premier temps, nous avons à l'aide d'un cutter, découpé environ 5 cm de largeur du carton afin qu'il puisse rentrer dans la machine à découpe laser.

- Après nous avons utilisé un schéma qui nous est fourni sur uncloud afin d'indiquer le chemin à prendre par le laser. Nous avons ensuite réglé la vitesse du laser à 20 et laissé la puissance du laser à 100 %.

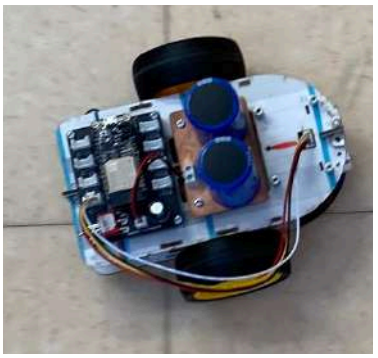
- Nous avons ensuite envoyé l'information à l'automate qui a fait la découpe.

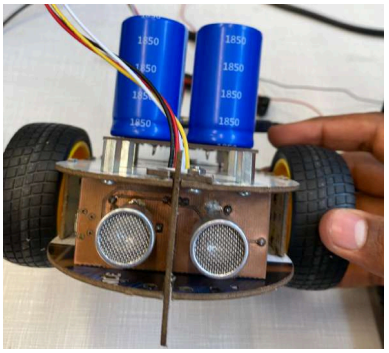
- Nous nous retrouvons ainsi à la fin avec les pièces suivantes à monter ensemble:
    1. Il y avait en tout 15 vis et boulons.
    2. Nous avons dû percer de nouveaux trous pour la fixations de la carte avec les condensateurs car la distance des trous était décalée de 1 cm.





- Une fois monté le robot ressemble à ça:

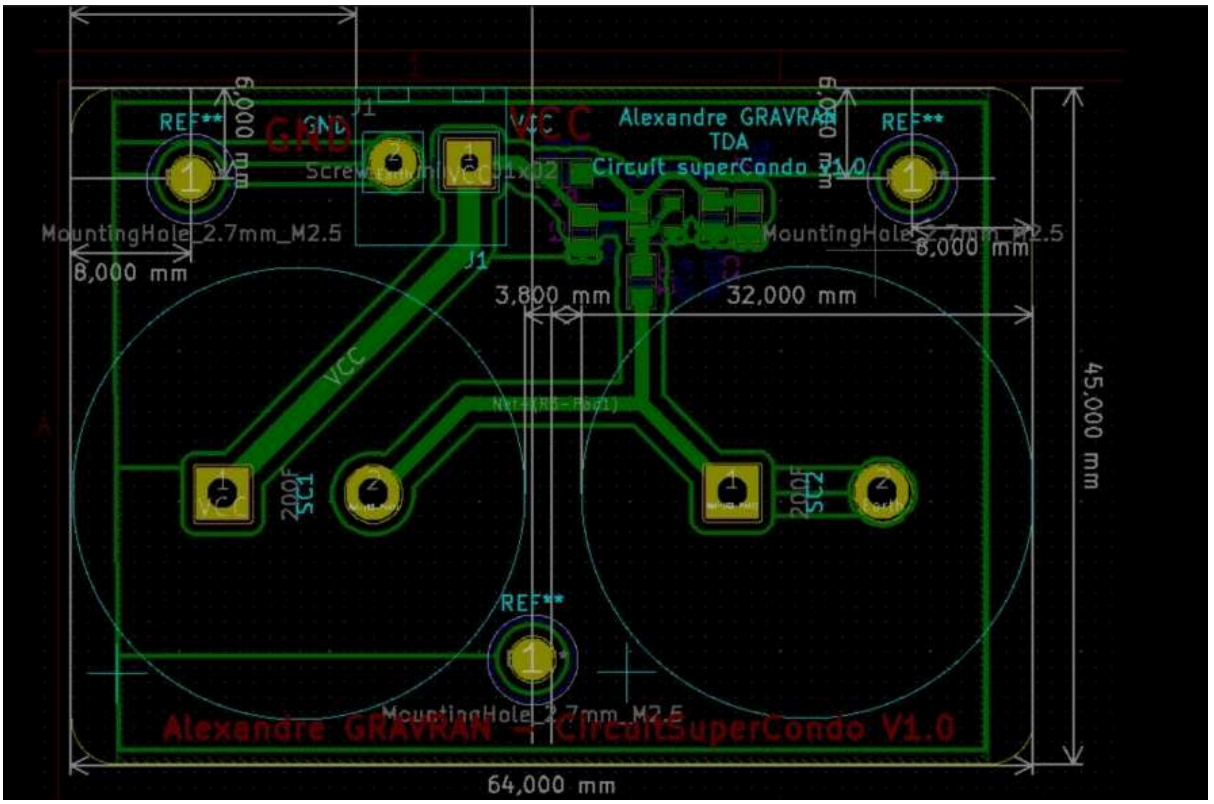




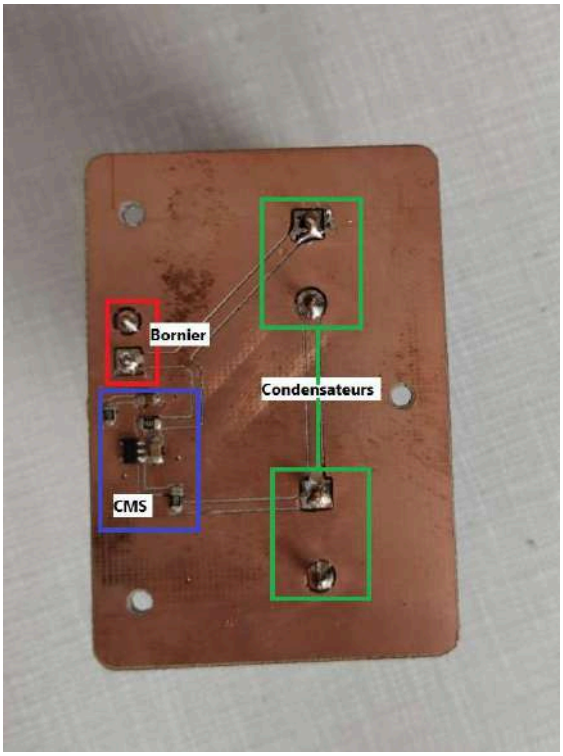
4 Alimentation du robot à l'aide de supercondensateur

Afin d'alimenter le robot, l'IUT à décidé de nous faire travailler sur des supercondensateurs pour l'alimentation des robots.
Ils ont une autonomie d'environ 4 min et sont avantageux car "les supercondensateurs sur les batteries Lithium-Ion tiennent également à leur efficacité (98% contre entre 75 et 90%, grâce à une perte par chaleur minime), leur cyclabilité (nombre de cycle charge-décharge, compris entre 500 000 et 20 millions, contre environ 1 000 pour une batterie Lithium-Ion), une plus grande amplitude de température pour la charge et la décharge (de -40°C à +65°C), et leur sûreté. Avec un supercondensateur, aucun problème de décharge profonde, de risque de surcharge, d'emballement thermique ou d'explosion – contrairement à une batterie Lithium-Ion." (source: Supercondensateurs Lithium-Ion : sont-ils l'avenir du stockage d'électricité ? ,https://les-smartgrids.fr/supercondensateurs-lithium-ion-avenir-stockage/).

- Sur Kicad nous avons donc dessiné la carte qui permettra plus tard d'alimenter les supercondensateurs, voir schéma ci-dessous(le schéma est celui d'Alexandre Gravran car je ne retrouve plus mon schéma en cherchant sur l'accès nomade).



- Une fois imprimé et monté nous avons la carte suivante :



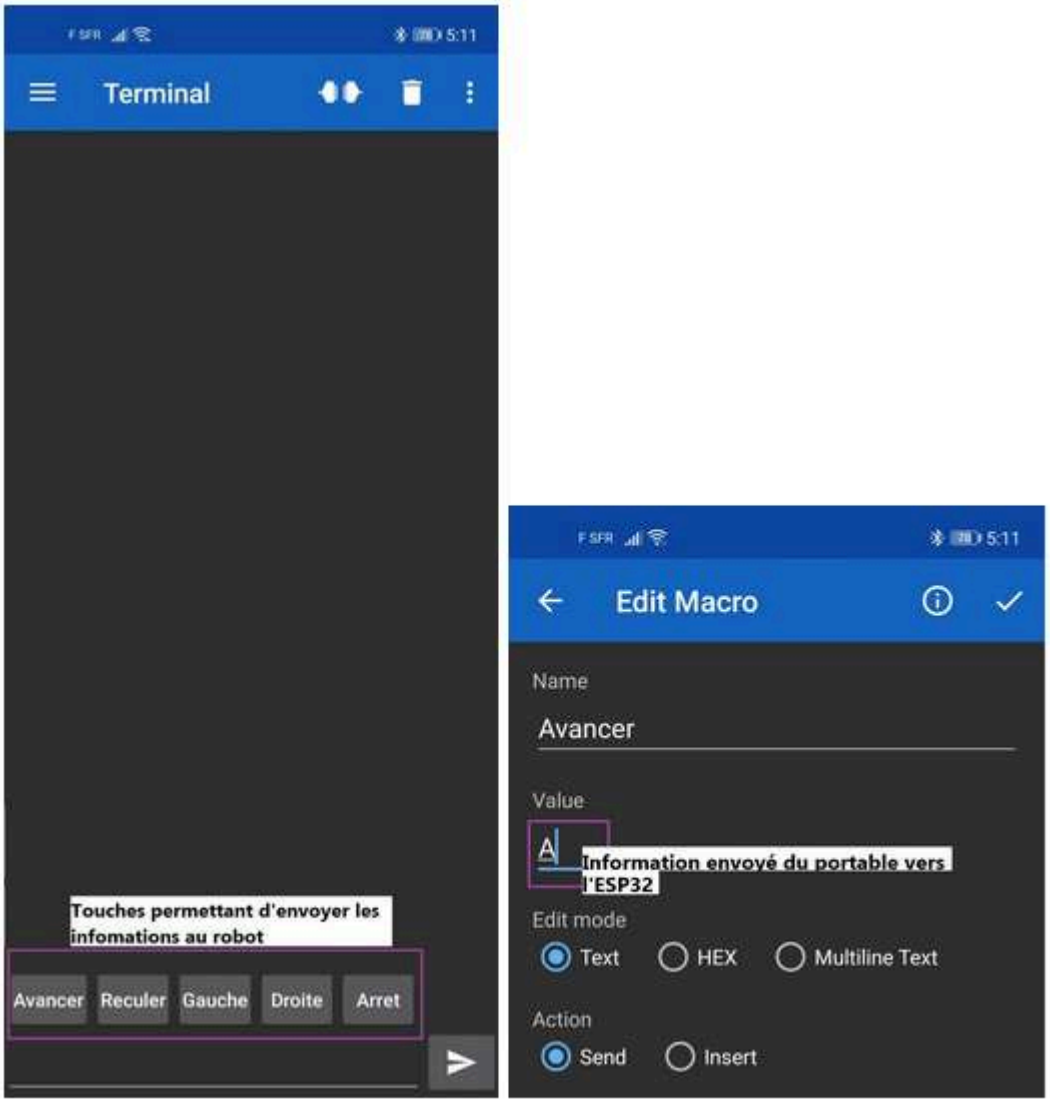
- Les supercondensateurs sont rechargés jusqu'à 4.2V à 3A à l'aide d'un chargeur à supercondensateur .

5 Programmation avec le Framework Arduino

5.1 Déplacement manuel

Pour commencer à programmer le robot, nous avons travaillé sur le déplacement manuel du robot. A l'aide de notre portable connecté en Bluetooth avec l'ESP 32 (application Serial Bluetooth) nous envoyons les informations de déplacement au robot.

Copie d'écran du portable permettant de contrôler le robot:



Programme permettant le contrôle manuel:

```
        //Contrôle de la direction du robot en fonction de
l'information envoyé par le portable
        if (sens == 'A')
        {
          avancer(); //appel la fonction gérant les moteurs
        }

        if (sens == 'R')
        {
          reculer();
        }

        if (sens == 'D')
        {
          droite();
        }

        if (sens == 'G')
        {
          gauche();
        }

        if (sens == 'S')
        {
          arret();
        }
      }

      void avancer(void)
      {
```

```
	digitalWrite(AD,1);
	digitalWrite(AG,1);
	digitalWrite(RD,0);
	digitalWrite(RG,0);
}

void reculer(void)
{
	digitalWrite(AD,0);
	digitalWrite(AG,0);
	digitalWrite(RD,1);
	digitalWrite(RG,1);
}


void droite(void)
{
	digitalWrite(AD,0);
	digitalWrite(RG,0);
	digitalWrite(AG,1);
	digitalWrite(RD,1);
	delay(75);
	digitalWrite(AG,0);
	digitalWrite(RD,0);
}


void gauche(void)
{
	digitalWrite(AD,1);
	digitalWrite(RG,1);
	digitalWrite(AG,0);
	digitalWrite(RD,0);
	delay(75);
	digitalWrite(AD,0);
	digitalWrite(RG,0);
}

void arret(void)
{
	digitalWrite(AD,0);
	digitalWrite(AG,0);
```

```
        digitalWrite(RD,0);
        digitalWrite(RG,0);
    }
```

Je n'ai pas utilisé le PWR pour gérer la puissance des moteurs, car après avoir fait des tests avec Guillaume Lammour qui fait lui aussi la fonction FP2. Nous avons remarqué qu'il était compliqué de gérer correctement la puissance pour que le robot aille tout droit. De plus, selon la puissance, les moteurs ne fonctionnaient plus. J'ai donc décidé de ne pas intégrer cette fonction dans mon programme et de laisser le robot rouler à sa vitesse maximale.

5.2 Déplacement automatique

Afin que le robot se déplace automatiquement j'ai dû attendre que Pierre finisse le sonar. Une fois finis je l'ai intégré au robot et ai programmer l'ESP32 de tel sorte à ce que le robot tourne vers la gauche puis avance à chaque fois qu'il croise un obstacle.

Le programme est donc le suivant:

```
    float temps = pulseIn(ECHO_PIN,HIGH); //Temps que met l'ultrason
pour partir et revenir vers le robot
    float distance_mm = temps / 2.0 *0.34; //Calcul la distance du
robot par rapport à un obstacle en fonction de la période de l'echo

     /*Évitement des obstacles situés à 25cm du robot en tournant à
gauche puis en avançant.
     if(distance_mm < 250)
        {
         gauche();
         delay(70);
         avancer();
         delay(100);
         }
```

5.3 Déplacement et recherche de la balise

La fonction travaillant sur la balise n'ayant pas pu finir leur programme à cause de nombreux problèmes, j'ai décidé de faire la programmation au cas où nous aurions eu la fonction balise et avons imiter cette fonction en définissant une des broches en input_pullup afin d'avoir un 1 (balise détecté) ou un 0 (balise non détecté) à son entré comme si nous aurions eu la balise de connecté. Malheureusement, je n'ai pas eu le temps de tester ma fonction ainsi, je dépose juste le code avec les commentaires.

```
#define balise A4 //La broche A4 reçoit un 1 (balise détecté et en face) ou un 0 ( la balise n'est pas en face)

/*Évitement des obstacles situés à 25cm du robot en tournant à gauche puis ensuite en avançant.
 * Après avoir évité un obstacle, le robot recherche la balise afin d'être sûr d'aller dans le bon sens.
 */
if(distance_mm < 250)
{
 gauche();
 delay(70);
 avancer();
 delay(100);
 Detection_balise(balise);
}

//la fonction fait tourner le robot à droite tant qu'il n'a pas trouver la balise
void Detection_balise(int balise)
```

```
{
 while(balise == 0)
 {
     droite();
 }
}
```

Programme complet:

```
#include "BluetoothSerial.h"
#include <stdint.h>
#include <stdbool.h>
#include "esp_err.h"
#include "esp_intr_alloc.h"
#include "soc/soc.h"
#include "driver/ledc.h"
#include "driver/gpio.h"
#include "driver/periph_ctrl.h"

#if !defined(CONFIG_BT_ENABLED) ||
!defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run 'make menuconfig' to
and enable it
#endif

#define RD A7 //La broche A7 permet de faire tourner le moteur
droit dans le sens trigonométrique
#define AD A6 //La broche A6 permet de faire tourner le moteur
droit dans le sens horaire
#define RG A9 //La broche A9 permet de faire tourner le moteur
droit dans le sens trigonométrique
#define AG A8 //La broche A8 permet de faire tourner le moteur
droit dans le sens horaire
#define balise A4 //La broche A4 reçoit un 1 (balise détecté et
en face) ou un 0 ( la balise n'est pas en face)
```

```
      const byte TRIGGER_PIN = 17; //Broche envoyant l'ultra-son
      const byte ECHO_PIN = 16;    //Broche recevant l'echo de
l'ultra-son
      unsigned char sens; //Variable permettant de contrôler le sens du
robot
      BluetoothSerial SerialBT;

      void setup()
      {
       Serial.begin(115200); //Vitesse de transmission
       SerialBT.begin("Pastis"); //Nom du robot
       Serial.println("Device on, pair it"); //Informe si le robot est
connecté ou non au portable

       //Initialisation des broches permettant de contrôler les moteurs
en sorti
       pinMode(RD, OUTPUT);
       pinMode(AD, OUTPUT);
       pinMode(AG, OUTPUT);
       pinMode(RG, OUTPUT);
       pinMode(TRIGGER_PIN,OUTPUT);//Initialisation de la broche en
sortie afin d'envoyer l'ultrason
       pinMode(ECHO_PIN,INPUT);//Initialisation de la broche en entrés
afin de recevoir l'ultrason
       pinMode(balise,INPUT_PULLUP);//Permet de mettre à 1 ou à 0 la
broche
      }

      void loop()
      {
         float temps = pulseIn(ECHO_PIN,HIGH); //Temps que met
l'ultrason pour partir et revenir vers le robot
         float distance_mm = temps / 2.0 *0.34; //Calcul la distance du
robot par rapport à un obstacle en fonction de la période de l'echo

         if(Serial.available()){
           SerialBT.write(SerialBT.read());//Permet de recevoir les
informations envoyé par le robot
           }

         if(SerialBT.available())
         {
```

```
          sens = SerialBT.read(); //Sens prendra la valeur envoyé par
le portable
          Serial.write(sens); //Permet de voir sur le portable la
valeur envoyé
          Serial.println(sens);//Permet de voir sur la console la
valeur envoyé
          Serial.println("/n",distance_mm);//Permet de voir sur la
console la distance en temps réel
          }

          /*Évitement des obstacles situés à 25cm du robot en tournant
à gauche puis ensuite en avançant.
           * Après avoir évité un obstacle, le robot recherche la
balise afin d'être sûr d'aller dans le bon sens.
           */
          if(distance_mm < 250)
          {
           gauche();
           delay(70);
           avancer();
           delay(100);
           Detection_balise(balise);
          }

         //Contrôle de la direction du robot en fonction de
l'information envoyé par le portable
         if (sens == 'A')
         {
           avancer();//appel la fonction gérant les moteurs

         }

         if (sens == 'R')
         {
           reculer();
         }

         if (sens == 'D')
         {
           droite();
         }
```

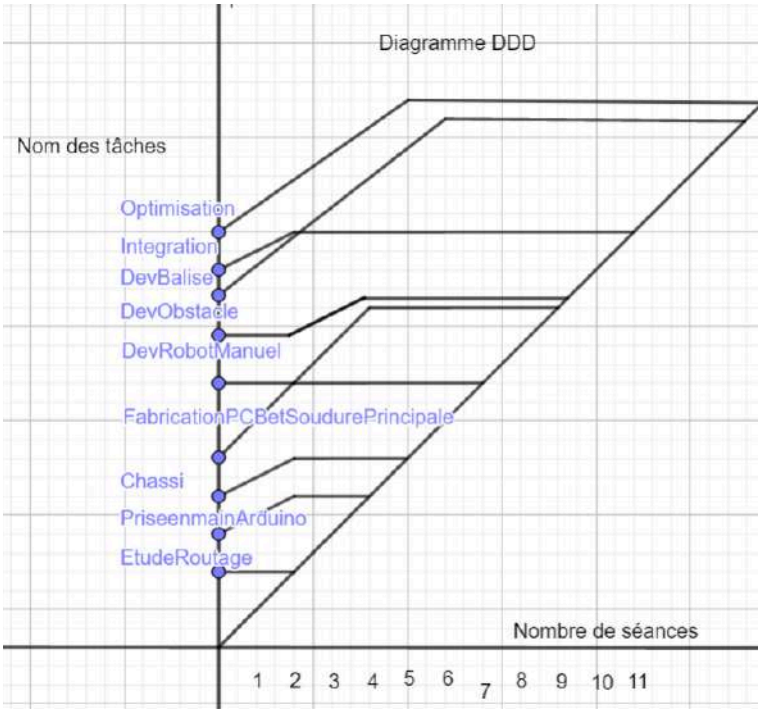
```
        if (sens == 'G')
        {
          gauche();
        }

        if (sens == 'S')
        {
          arret();
        }
      }

      //la fonction fait tourner le robot à droite tant qu'il n'a pas
trouver la balise
      void Detection_balise(int balise)
      {
       while(balise == 0)
       {
           droite();
       }
      }

      //fonction permettant de contrôler la direction du robot
      void avancer(void)
      {
           digitalWrite(AD,1);
           digitalWrite(AG,1);
           digitalWrite(RD,0);
           digitalWrite(RG,0);
      }

      void reculer(void)
      {
           digitalWrite(AD,0);
           digitalWrite(AG,0);
           digitalWrite(RD,1);
           digitalWrite(RG,1);
      }


      void droite(void)
      {
           digitalWrite(AD,0);
```

```
	digitalWrite(RG,0);
	digitalWrite(AG,1);
	digitalWrite(RD,1);
	delay(75);
	digitalWrite(AG,0);
	digitalWrite(RD,0);
}


void gauche(void)
{
	digitalWrite(AD,1);
	digitalWrite(RG,1);
	digitalWrite(AG,0);
	digitalWrite(RD,0);
	delay(75);
	digitalWrite(AD,0);
	digitalWrite(RG,0);
}

void arret(void)
{
	digitalWrite(AD,0);
	digitalWrite(AG,0);
	digitalWrite(RD,0);
	digitalWrite(RG,0);
}
```

L'emploi du temps :

Séance	Groupes A/B	Groupes C/D	Objectifs
1	15	16	Etude & Routage
2	15	16	Programmation
3	vacances	vacances	
4	21	20	Fabrication des PCB
5	21	20	Fabrication du chassis
6	23	22	Programmation
7	23	22	Tests & Validations
8	24	23	Programmation
9	24	23	Programmation / Intégration au robot
10	25	24	Programmation / Intégration au robot
11	25	24	Optimisation
coupe			

Notre groupe de TD n'a pas pu implémenter toutes les fonctions sur le robot dû à de nombreux problèmes cependant tous les programmes ont été terminés à peu près à temps (voir le DDR ci-dessous).

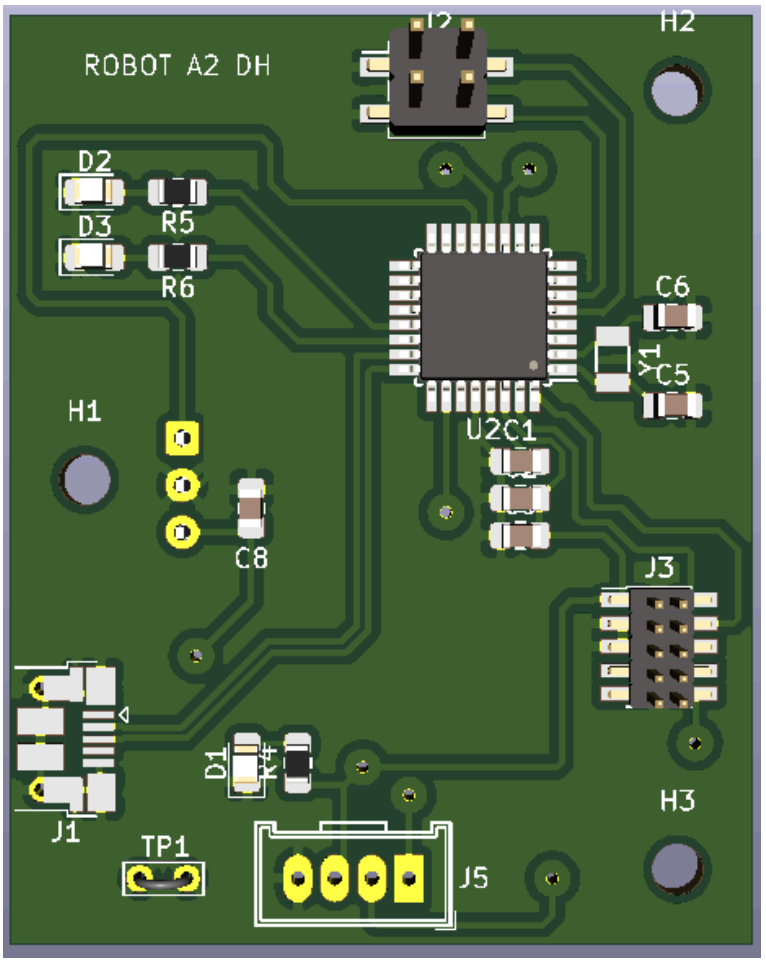


Conclusion:

La fonction FP2 a été intéressante, car nous avons eu à faire de multiples tâches différentes allant de la programmation à la découpe laser pour enfin faire du soudage. De plus, vu que je n'étais pas surchargée, j'ai pu aider à de multiples reprises mes camarades sur les autres fonctions avec la soudure afin de réparer leur carte (même des camarades qui n'étais pas dans mon groupe). J'ai aussi appris à gérer la puissance de moteur grâce à la fonction PWR.Le seul point frustrant est que nous n'avons pas pus mettre toutes les fonctions sur le robot afin qu'il fonctionne à 100 % automatiquement.

3. La fonction FP4 «balise»

(Dimitri Hamon)



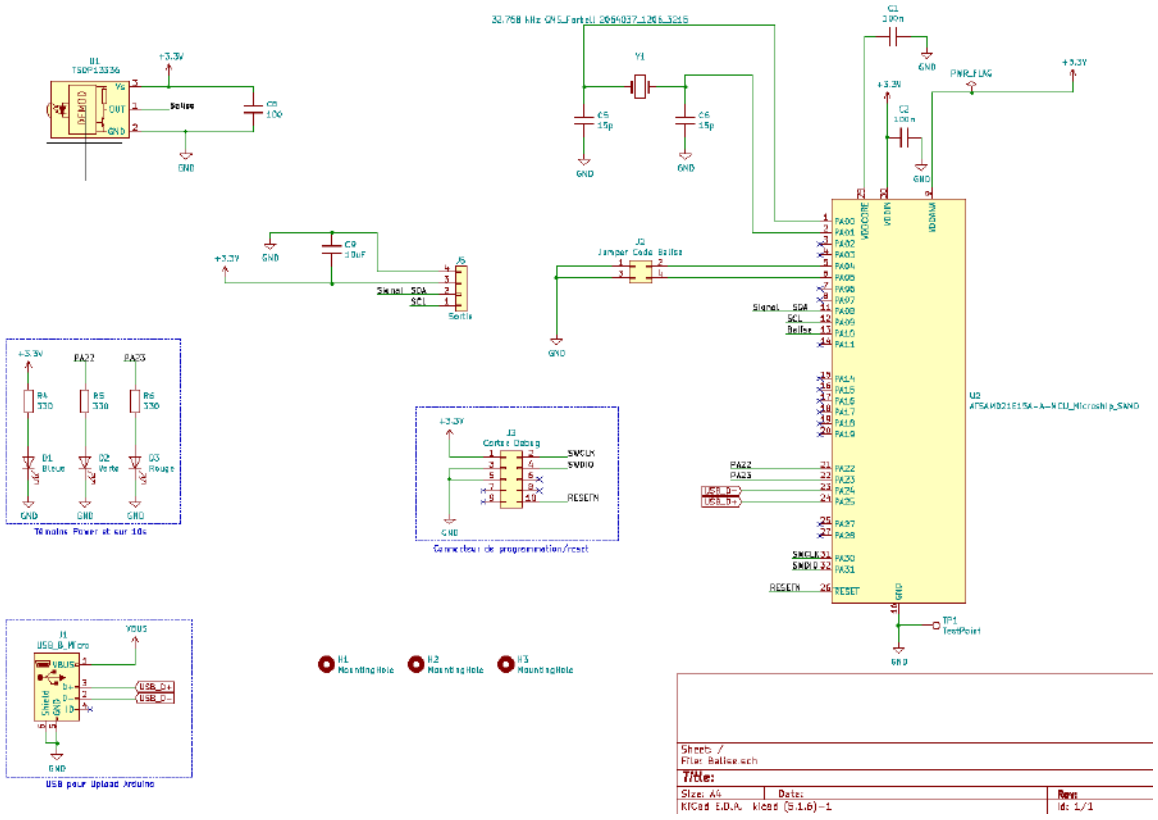
Le détecteur se compose de 2 parties:

- une partie mécanique intégrant le récepteur infrarouge placé en haut du mât présent au centre du robot (FS41)
- un circuit de décodage capable d'interpréter le code reçu par la diode démodulatrice (FS42).Les deux éléments sont reliés par un câble Grove ou un câble 3 fils en fonction du choix de connectique effectué par le groupe.

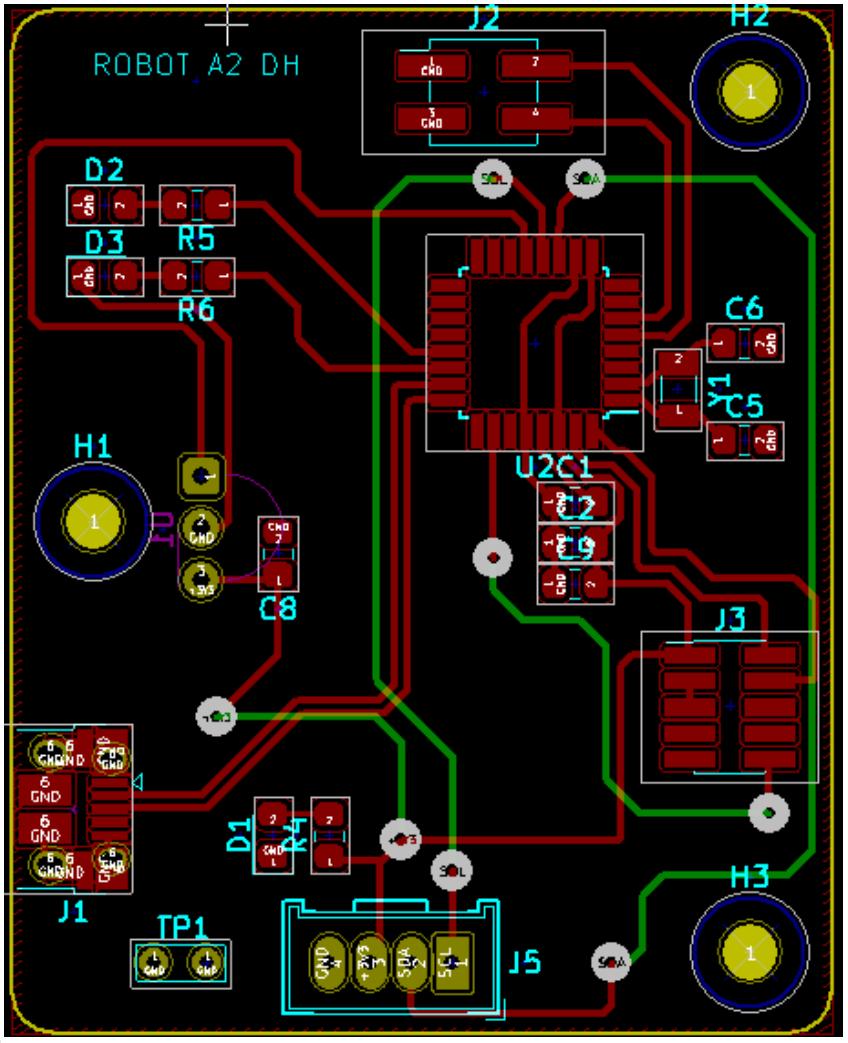
3.1 La réalisation du PCB

3.1.1 SAISIE DU SCHÉMA SOUS KICAD

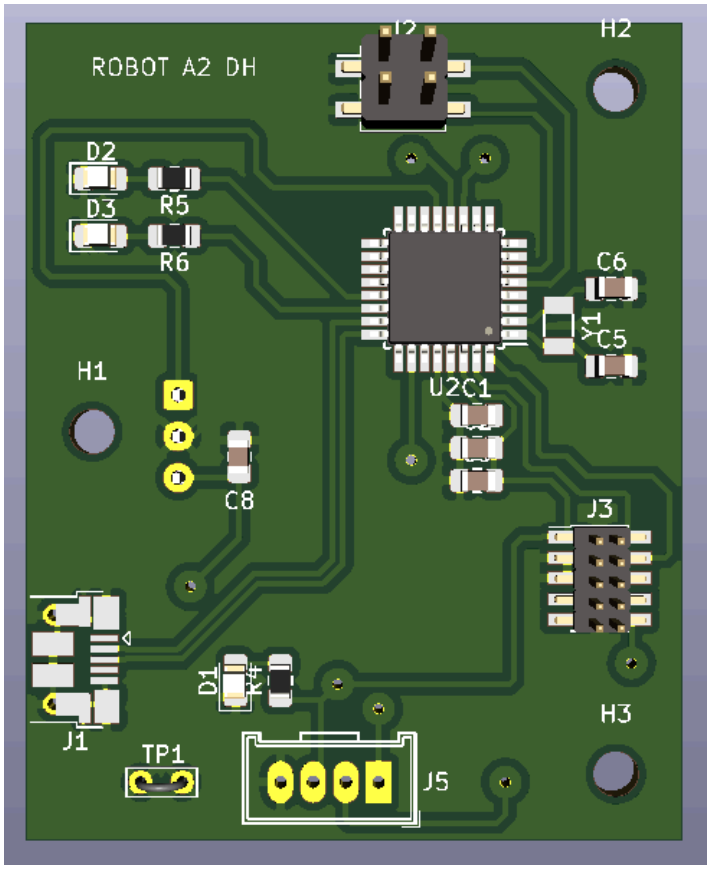
Pour la réalisation du PCB, j'ai tout d'abord saisi le schéma ci-dessous:

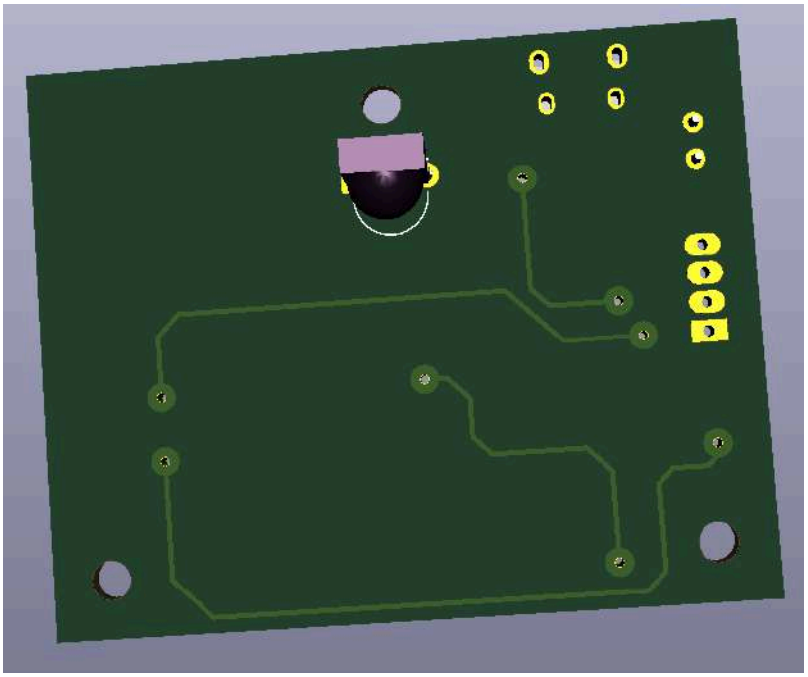


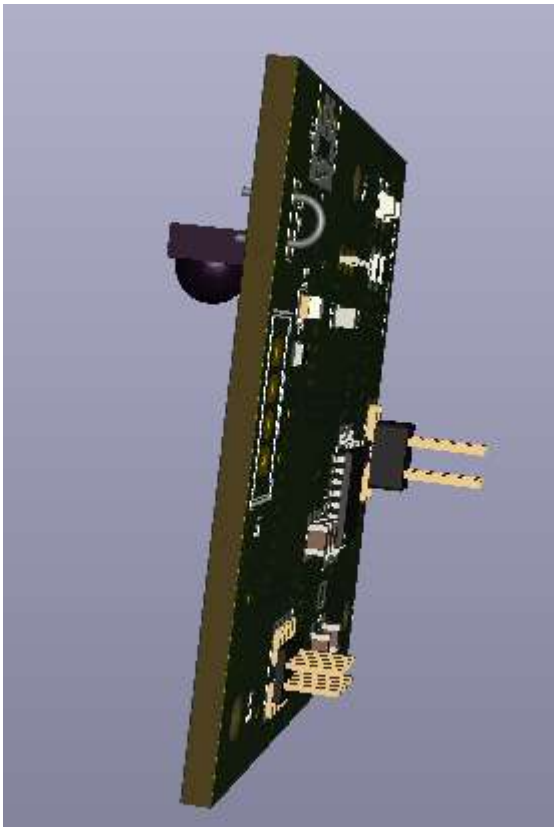
3.1.2 ROUTAGE


ROBOT A2 DH
J2
H2
D2
D3
R5
R6
C6
Y1
C5
H1
U2C1
C8
J3
J1
D1
R4
H3
TP1
J5
GND
+3V3
SDA
SCL

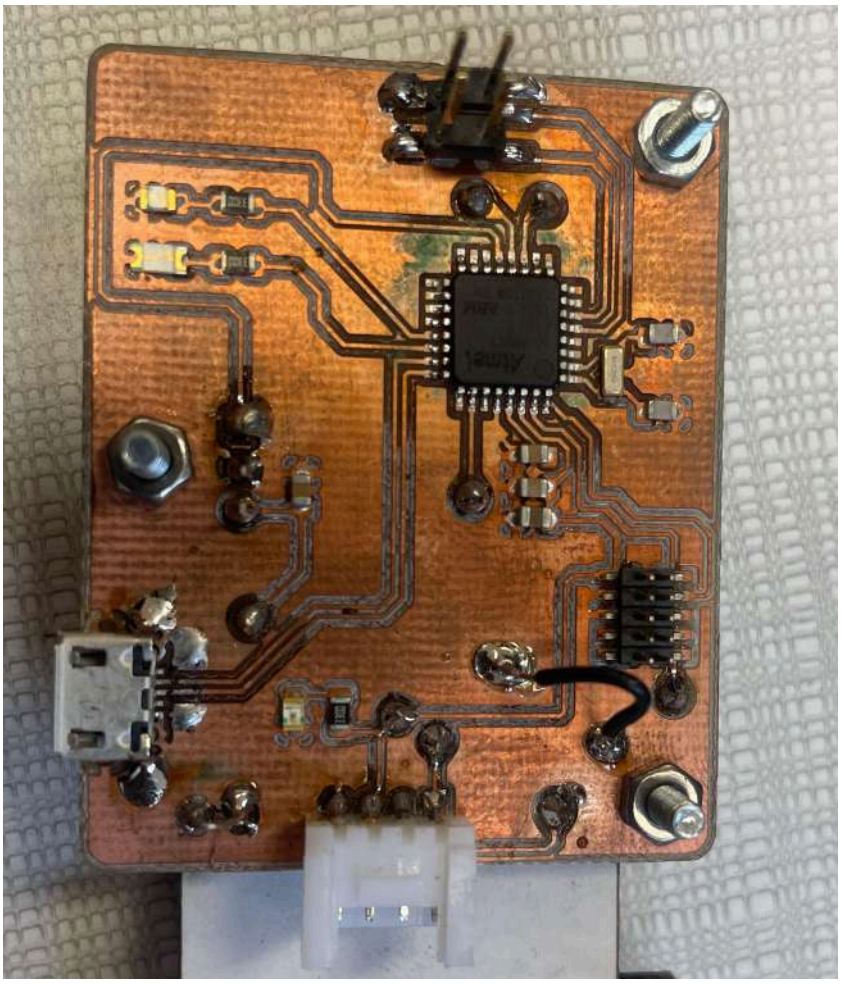
3.1.3 VUES 3D DE LA CARTE







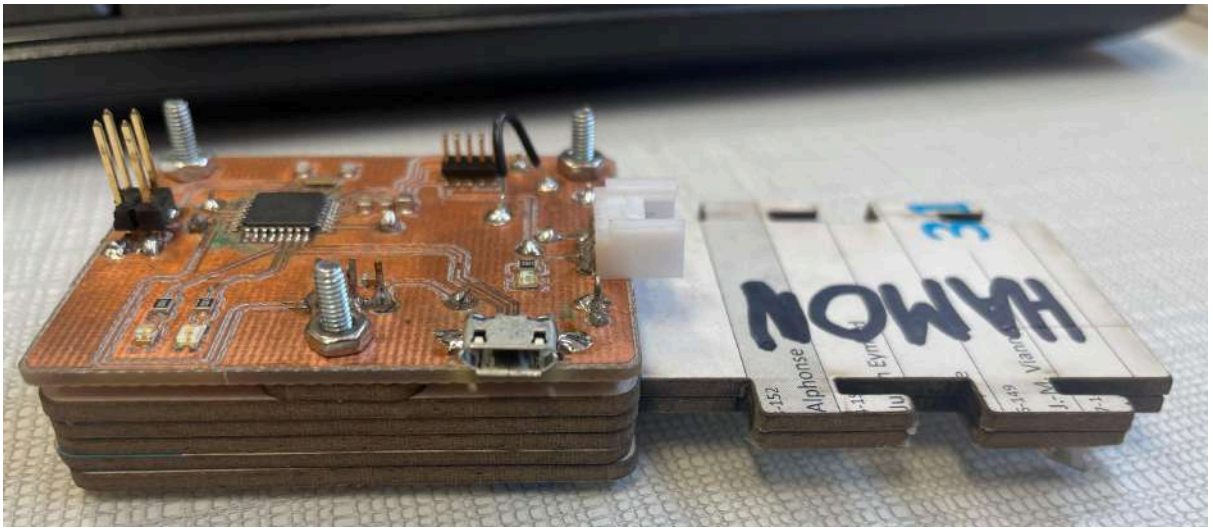
3.1.4 LISTE DU MATÉRIELS

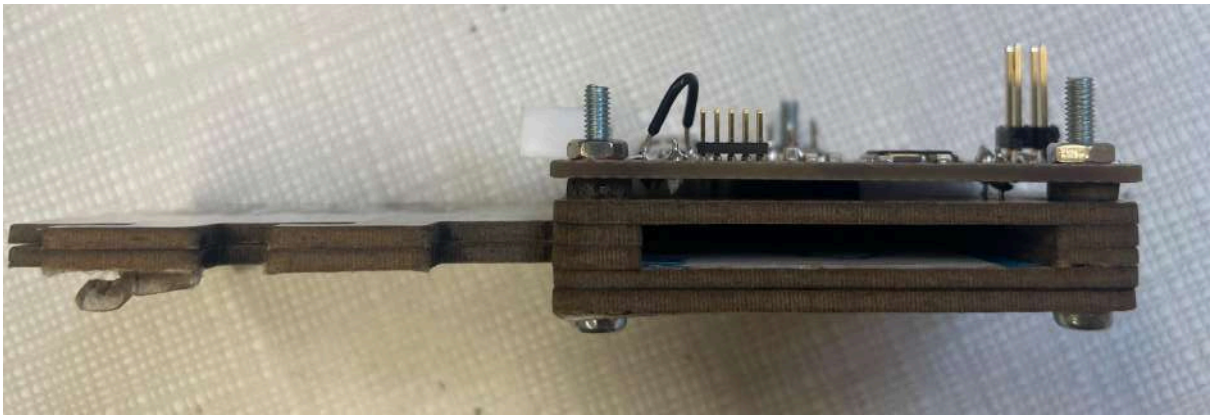


3.1.5 Placement des composants et assemblage de la carte

Concernant le placement des composants, j'ai mis de la pâte à braser, puis placer les composants. Par la suite, j'ai mis la carte au four thermique, pour que les composants s'accrochent.

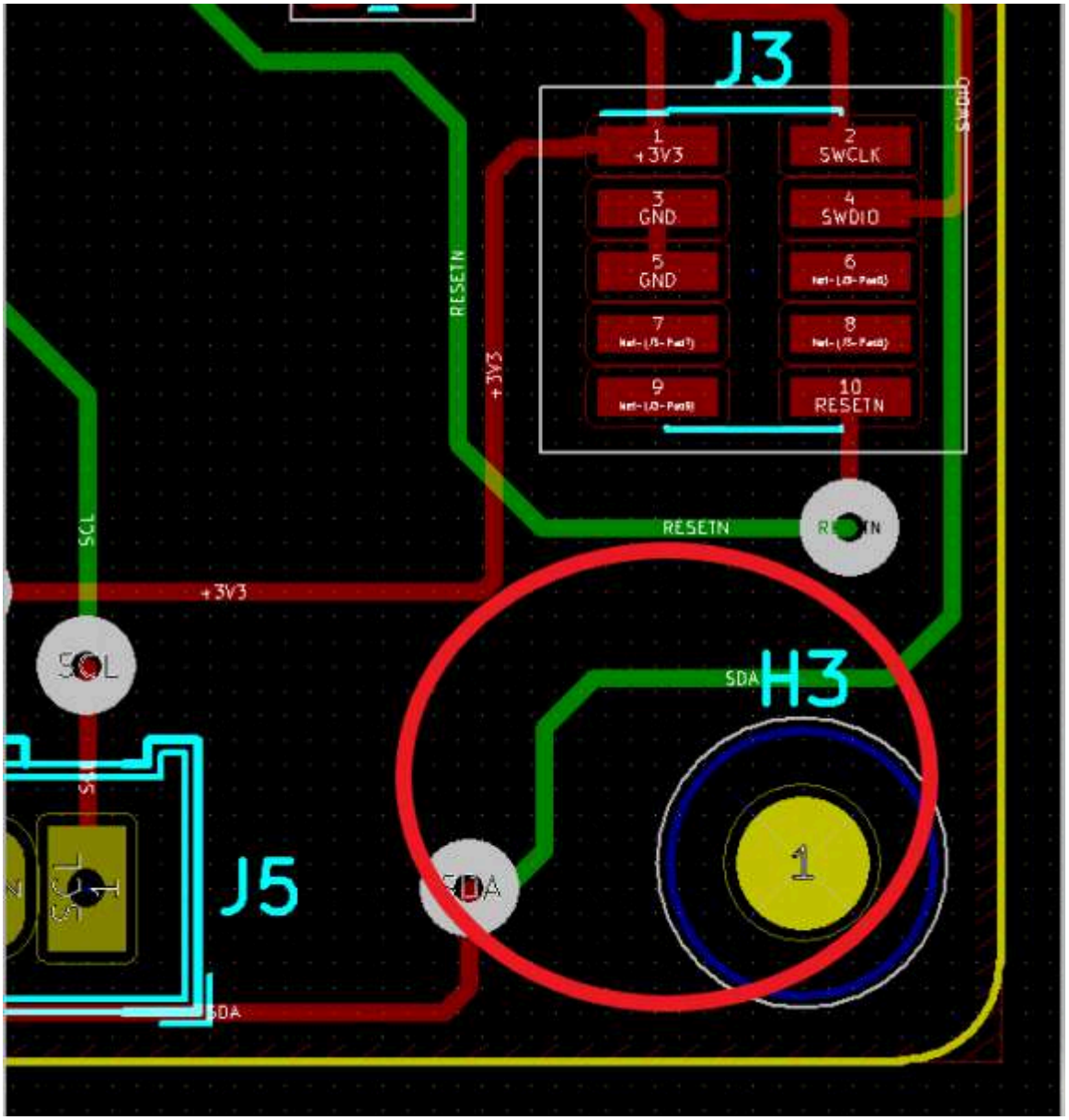
Avant de passer au test, il a fallu découper des morceaux de carton, grâce à la découpeuse laser, pour que la carte puisse être mise sur le robot.



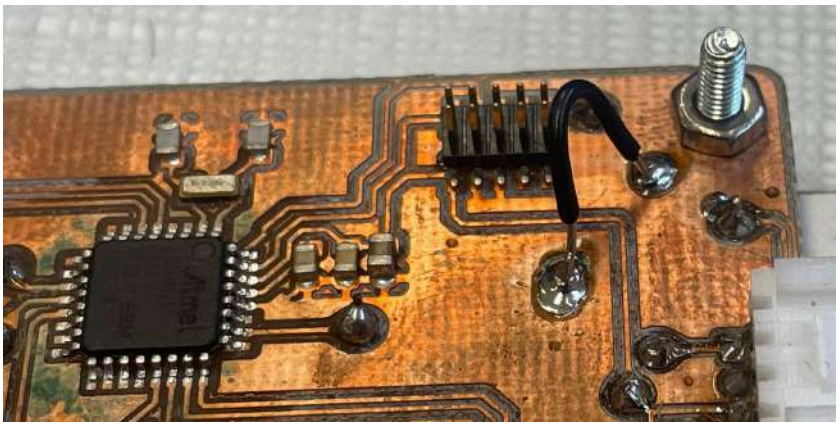


3.2 Premiers tests de la carte

Après avoir réalisé toute la carte, j'ai dû passer aux tests de continuité.
J'ai d'abord alimenté la carte avec du 3.3V : aucune leds ne s'allumaient.
En effet sur le routage (voir ci-dessous), le plan de masse n'était pas relié.



Pour remédier à ce problème, j'ai soudé un câble avec le plan de masse à côté (voir ci-dessous).



Ensuite, j'ai voulu passer à la programmation. Malheureusement, par manque d'attention, au lieu d'alimenter en 3.3V, j'ai envoyé du 9V.
Par conséquent, mon SAMD21 ne fonctionnait plus. J'ai dû le changer, en coupant les pattes, enlever la soudure, mettre du flux, puis le replacer et le souder à la main.

De plus, j'ai eu un problème avec mon connecteur micro-USB (comme beaucoup de personnes). Personnellement, il s'est seulement dessoudé.

3.3 Programmation et simulations

Tout d'abord, j'ai voulu testé, par un simple programme, si les LEDS clignotaient.

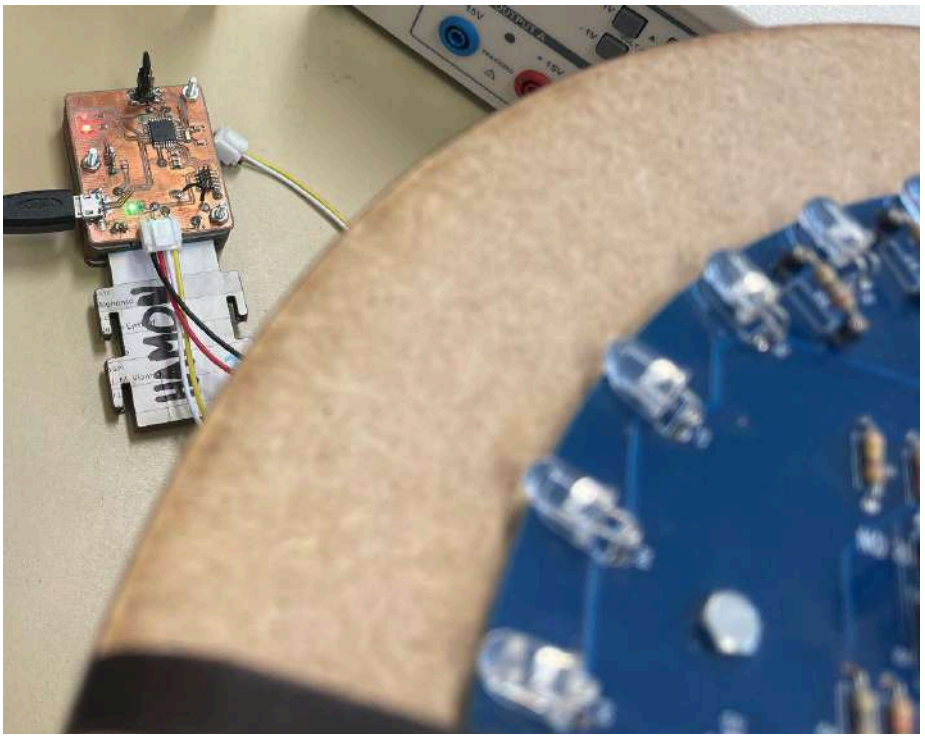
```
#define LEDB 23
#define LEDR 22

void setup() {
  pinMode(22, OUTPUT);
  pinMode(23, OUTPUT);
}

void loop() {
  digitalWrite(LEDB, HIGH);
  delay(100);
  digitalWrite(LEDB, LOW);
  delay(100);
  digitalWrite(LEDR, HIGH);
  delay(100);
  digitalWrite(LEDR, LOW);
  delay(100);
}
```

Mes LEDS clignotaient bien. Par la suite, j'ai réalisé un programme tout aussi simple, pour savoir si ma carte recevait bien quelque chose de la part de la balise.

```
#define IR 10
#define LEDV 22
#define LEDR 23

void setup() {
  Serial.begin(9600);
  pinMode(IR, INPUT);
  pinMode(LEDR,OUTPUT);
  pinMode(LEDV,OUTPUT);
  digitalWrite(LEDR, LOW);
  digitalWrite(LEDV, LOW);
}


void loop()
{
 Serial.println(digitalRead(IR));

  if (digitalRead(IR) != 1)
  {
  digitalWrite(LEDR, LOW);
  digitalWrite(LEDV, HIGH);
  delay(1000);
  Serial.println(digitalRead(IR));
  }
  digitalWrite(LEDV, LOW);
  digitalWrite(LEDR, HIGH);
}
```

Ma carte recevait bien un signal. En effet les LEDS clignotaient bien.

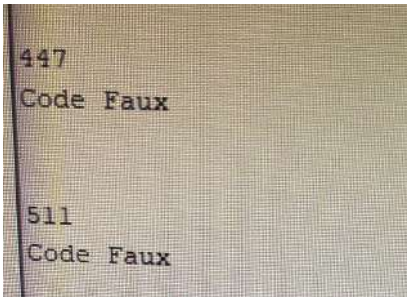
Par la suite, créer un programme pour la balise était compliqué. En s'aidant avec les camarades de la fonction FP4, nous sommes arrivés au résultat suivant:



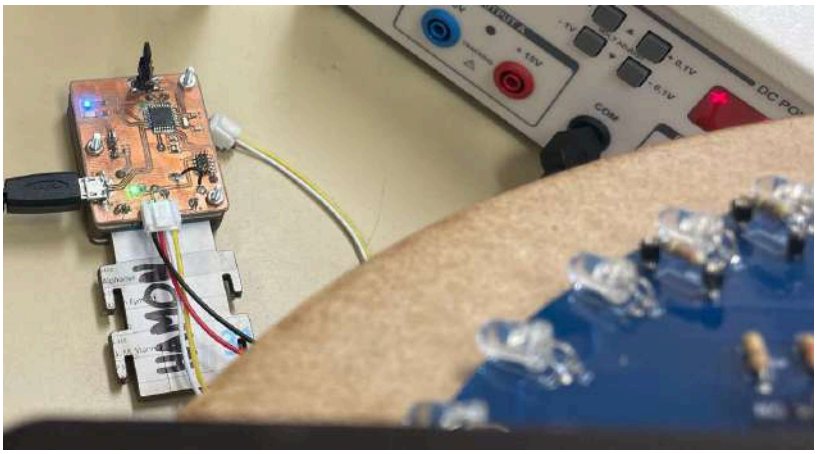
Nous avons réglé un code sur la balise émettrice.
Notre carte devait le recevoir, et analyser si ce dernier était correct ou non.
Pour la simuler, nous devions placer des cavaliers (voir photo ci-dessus).

J'ai choisi la balise numéro 3 pour réaliser mes tests:

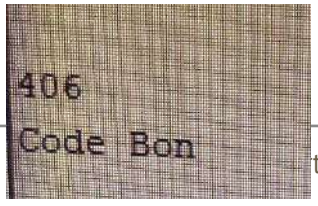
```
// Balise 3 : 1 0 1 1 0 => 0110010110 => on recoit : 0110100110
#define BALISE3 406

if(cavalier5 ==1 && cavalier6==1)
  {
   codebon=BALISE3;
  }
  else
  {
   etat=ERREUR;
  }
```



Le code fonctionnait bien. Comme nous pouvons le remarquer, le moniteur série affichait "code faux", et la LED rouge était allumée.



Ci-dessous, j'ai réglé la balise émettrice sur le bon code (406 dans ce cas)



Le moniteur série affiche bien “code bon”, et la LED verte est allumée.
Nous pouvons conclure que le programme fonctionne bien.

Malheureusement, par manque de temps, l’implantation sur le robot n’a pu être possible.

3.4 Conclusion

Ce module ER2 m’a beaucoup plu. La réalisation d’un projet de groupe est vraiment quelque chose que j’apprécie. Cependant, je trouve que nous n’avons pas assez travaillé ensemble, car le travail de chaque fonction n’étant pas terminé, nous n’avons pas pu assembler tout le robot.

En début de projet, j’ai avancé très vite. Grâce au module de AA, la réalisation du PCB sous Kicad était facile. Par la suite, la partie que j’ai préféré, est celle du soudage de la carte. Malgré les problèmes rencontrés, tel que l’oubli d’un plan de masse, le SAMD21 à changer, je n’ai pas baissé les bras.

En fin de projet, l’avancement était plus lent. En effet, la saisie du programme était vraiment compliquée car nous n’avions quasiment aucune information de départ du programme. En s’aidant, nous sommes arrivés à un résultat assez satisfaisant.

Pour conclure, les modules d’ER1, de CP et de PT nous ont préparés à réaliser un mini projet. Ces cours m’ont vraiment aidé. Je tiens à remercier les professeurs qui ont fait leur maximum pour nous dépanner. Je suis prêt pour le module d’ER3, lors de ma seconde année.

3.5 Annexe

```
#define PIN_IR 10          //liaisons cartes
#define PIN_SORTIE 23       //pin Rouge
#define TIMEOUT 12192      //
#define PERIODE_BIT 2000     //
#define T_ERREUR 300       //
#define NB_BITS_MAX 5      //
#define DELAI_SORTIE 133333    //
#define now (micros())       //
// Codes des balises
```

```
// Balise 1 : 1 0 1 0 1 => 0110011001 => on reçoit :
1001100110(accumulés dans l'autre sens)
#define BALISE1 409
// Balise 2 : 1 1 1 0 0 => 0101011010 => on recoit : 0101101010
#define BALISE2 346
// Balise 3 : 1 0 1 1 0 => 0110010110 => on recoit : 0110100110
#define BALISE3 406
// Balise 4 : 1 1 0 1 1 => 0101100101 => on recoit : 1010011010
#define BALISE4 357


const unsigned int balises[4] = {BALISE1, BALISE2, BALISE3, BALISE4};

const int CODE_BALISE = B0100;

typedef enum {ATTENTE, RECEPTION, RECEPTION_OK, ERREUR} tEtat;
typedef enum {OK, FRONT_MILIEU, FRONT_DEBUT, FRONT_OUT} tErreur;
#define TRACE_SIZE 20

struct tTrace {
 long time;
 char front;
};
tTrace trace[TRACE_SIZE];
int pTrace = 0;


void setup()
{
 pinMode(22, OUTPUT); //pin 22 led verte
 pinMode(PIN_SORTIE, OUTPUT); //pin 23 led Rouge
 pinMode(PIN_IR, INPUT); //pin 13 Balise
 //for (int i = 4; i < 8; i++) pinMode(i, INPUT_PULLUP); //pin 4 à 7
 pinMode(4, INPUT_PULLUP);
 pinMode(5, INPUT_PULLUP);
 Serial.begin(9600);
 //digitalWrite(4,HIGH);
 //digitalWrite(5,HIGH);
}

void delayUs(long us)
{
```

```
 long fin = us + micros();
 while (micros() < fin);
}

int code[4];


void loop()
{
 int cavalier5 = digitalRead(4);
 int cavalier6 = digitalRead(5);
 static tEtat etat = ATTENTE;
 static bool in = true;
 static bool lastIn = true;
 static bool frontMontant = false;
 static bool frontDescendant = false;
 static unsigned int dataRecue = 0;
 static unsigned int codebon = 0;
 static char nbDemiBits = 0;
 // Lecture des entrées
 lastIn = in;
 in = digitalRead(PIN_IR);
 // Nature du front
 frontMontant = ((lastIn == false) & (in == true));
 frontDescendant = ((lastIn == true) & (in == false));
 // Gestion de la machine à états
 switch(etat) {
   case ATTENTE :
   if (frontDescendant) {
     dataRecue = 0;
     nbDemiBits = 1;
     etat = RECEPTION;
     delayUs(1500);
 //  Serial.println("Attente");
   }
   break;

   case RECEPTION :
   dataRecue = (dataRecue << 1) + in;
   nbDemiBits++;
   if (nbDemiBits == 10) {
    // Serial.println("Recption");
```

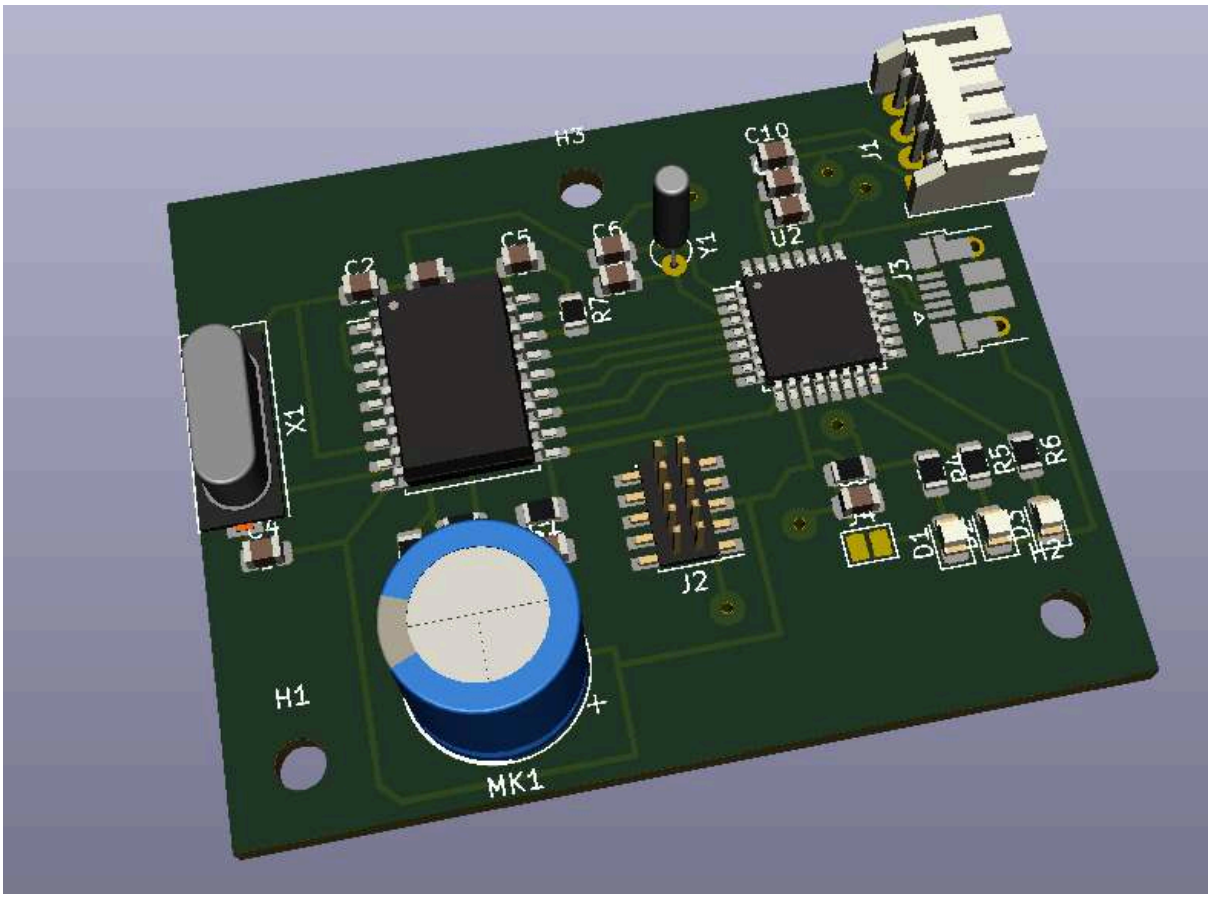
```
    etat = RECEPTION_OK;
  } else {
    delayUs(1000);
  }
  break;

  case RECEPTION_OK :

  // Comparaison avec le code de la balise
  // Lire les 2 cavaliers

  //Serial.print("\nCode carte= ");
  for(int k = 4; k < 8; k++)
  {
    //Serial.print(digitalRead(k));
    code[k-4]=digitalRead(k);
  }

  if(cavalier5 == 1 && cavalier6 == 0)
  {
    codebon=BALISE1;
  }
  else
  {
    etat=ERREUR;
  }

  if(cavalier5 == 0 && cavalier6 == 0)
  {
    codebon=BALISE2;
  }
  else
  {
    etat=ERREUR;
  }

  if(cavalier5 ==1 && cavalier6==1)
  {
    codebon=BALISE3;
  }
  else
  {
```

```
    etat=ERREUR;
  }

  if(cavalier5 ==0 && cavalier6==1)
  {
    codebon=BALISE4;
  }
  else
  {
    etat=ERREUR;
  }

  //codebon=BALISE4;

  //Serial.println(digitalRead(4));
  //Serial.println(digitalRead(5));
  Serial.print("\n");

  Serial.print("\n");
  Serial.print(dataRecue);
  digitalRead(PIN_IR);
// Serial.print(PIN_IR);
   if (dataRecue == codebon)
   {
      Serial.print("\nCode Bon\n");
      digitalWrite(22,HIGH);
      digitalWrite(PIN_SORTIE,LOW);
      digitalWrite(9,HIGH);
      etat=ATTENTE;
    }else
    {
     Serial.print("\nCode Faux\n");
     digitalWrite(PIN_SORTIE,HIGH);
     digitalWrite(22,LOW);
     digitalWrite(9,LOW);

      etat = ERREUR;
    }


  break;
  case ERREUR :
```

```
    etat = ATTENTE;
    break;
  }
}
```

FP7 DTMF

(Lucas ROUILLÉ)

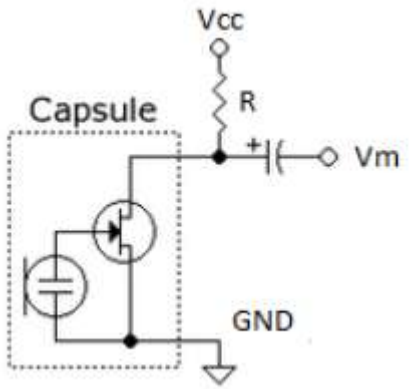


Introduction DMTF :

Nous réalisons aujourd'hui un robot dans le cadre du module ER2. Nous sommes répartis en groupe de 4 étudiants sur 4 fonctions différentes (sonar, Balise, DTMF et Gestion Robot). Personnellement, je m'occupe de la fonction FP7, autrement dit le DTMF. Cette fonction a pour but de faire démarrer le robot, nous comprendrons tout au long de ce compte-rendu comment marche un DTMF, mais également, nous réaliserons une carte qui devra s'incrémenter au robot Léo.

Question théorique :

C.1 Test du microphone Electret



Q1)

Tout d'abord, on lit la documentation.

Q2)

On nous demande de mettre en œuvre le micro afin de choisir une résistance adaptée pour que :
1V < Vm < 4V avec Vcc = 5V et une résistance R de 12 kΩ

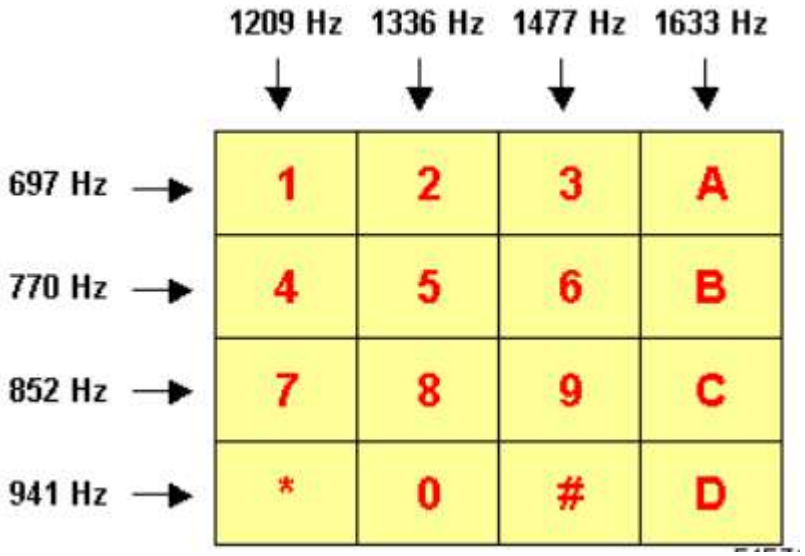
On relie ainsi le montage à un oscilloscope afin de mesurer Vm (ces mesures ont été réalisées avec Loan Martin.) :

Vm est bien compris entre : 1 V < Vm < 4V

C.2 MICROPHONE ÉLECTRET ET DTMF

Q3)

1- **La signalisation DTMF (*Dual-Tone Multi-Frequency*)** est un système par lequel des tonalités audibles sont utilisées pour représenter les boutons sur un clavier.



Q4)

Q5)

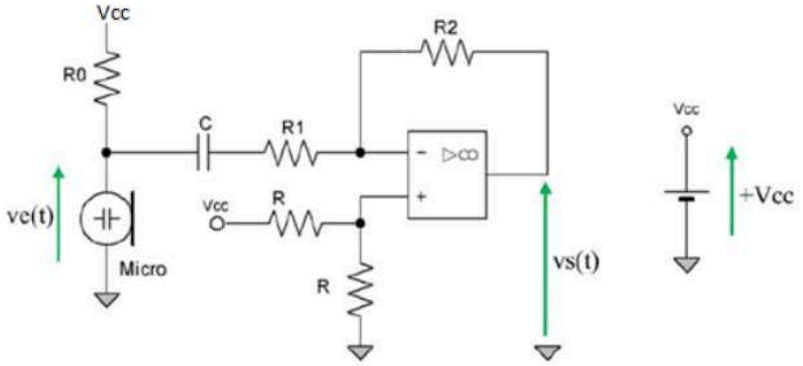
Les questions 4 et 5 n'ont malheureusement pas pu être traitées.

C.3 AMPLIFICATEUR MONO-TENSION

Le signal de ce microphone doit être envoyé au circuit décodeur DTMF HT9170 :

- la puissance en entrée du décodeur doit être comprise entre -29dBm et +1dBm.
- voir wikipedia pour plus de détails : https://fr.wikipedia.org/wiki/DBm

On veut une puissance de -10dBm à l'entrée du décodeur ce qui correspond à une tension efficace de 250mV. Comme la valeur efficace du signal Vm issu du microphone est inférieure à cette valeur, on met en œuvre un amplificateur mono-tension pour amplifier ce signal Vm.



Q6)

Comme vu en SE2, on a $Vs = A.(ve_2 - ve_1)$ donc, Vs =$\frac{R_2}{R_1}.(ve_2 - ve_1)$ainsi, le rapport d'amplification A=$\frac{R_2}{R_1}$. Dans l'énoncé on a :

|Zc|<<$\frac{R1}{10}$, on en déduit Zc =$\frac{1}{jCw}$ainsi, C >>$\frac{1}{\omega*\frac{R1}{10}}$avec $1000\pi < \omega < 4000\pi$

Q7) Pour calculer C on prendra ω= 3000π

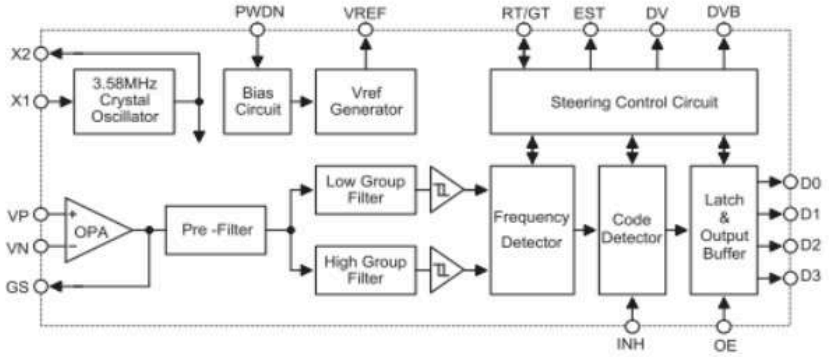
	Résistance R1= 1kΩ	Résistance R1= 10kΩ
Amplification A = 100	R2=100 kΩ | C = 1.06µF	R2=1 M | C = 0.106 µF
Amplification A = 1000	R2=1 MΩ | C = 1.06 µF	R2=10 MΩ|C = 0.106 µF

Q8) On a Vcc qui est appliqué sur l'entrée V_0 de l'AOP car le montage des deux résistances vont donner en tension d'entrée $V + = \frac{V_0}{2} = \frac{Vcc}{2}$. Cela permet de rehausser le signal de sortie pour obtenir un signal toujours positif et ainsi avoir un meilleur fonctionnement dans l'AOP.

D- Décodage DTMF

On utilise pour cette fonction le circuit DTMF HT9170. Ce circuit décode le signal DTMF et renvoie sur 4 bits (D0 à D3) la valeur du chiffre correspondant.
Le composant utilisé est en boîtier CMS 18 broches.

Schéma interne du circuit HT9170 :



On remarque que le composant intègre déjà un AOP (sigle OPA) afin de préamplifier le signal du micro.

Le tableau, situé page 10 de la documentation du composant, donne la valeur des codes obtenus en sortie sur D0 à D3 pour les différentes combinaisons de fréquence DTMF.

On précise que chaque donnée valide en sortie du récepteur DTMF est signalée par un niveau haut sur la broche DV.

Q9)

- PWDN : Active high. This enables the device to go into power down mode and inhibits the oscillator. This pin input is internally pulled down.

- INH : Logic high. This inhibits the detection of tones representing characters A, B, C And D. This pin input is internally pulled down.

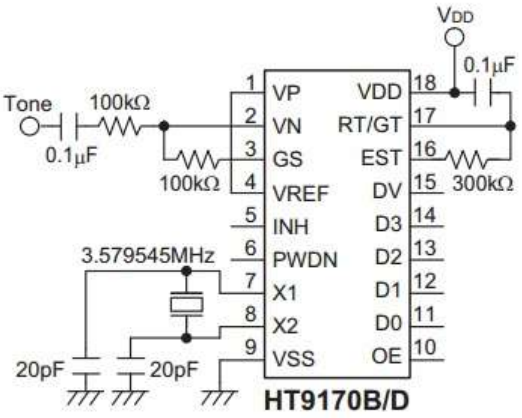
- OE : D0~D3 output enable, high active.

TABLE II
DTMF Data Output

Low group (Hz)	High group (Hz)	Digit	OE	D3	D2	D1	D0
697	1209	1	H	L	L	L	H
697	1336	2	H	L	L	H	L
697	1477	3	H	L	L	H	H
770	1209	4	H	L	H	L	L
770	1336	5	H	L	H	L	H
770	1477	6	H	L	H	H	L
852	1209	7	H	L	H	H	H
852	1336	8	H	H	L	L	L
852	1477	9	H	H	L	L	H
941	1336	0	H	H	L	H	L
941	1209	*	H	H	L	H	H
941	1477	#	H	H	H	L	L
697	1633	A	H	H	H	L	H
770	1633	B	H	H	H	H	L
852	1633	C	H	H	H	H	H
941	1633	D	H	L	L	L	L
—	—	ANY	L	Z	Z	Z	Z

Si on prend par exemple le chiffre 5, on a D3 au niveau bas (Low), D2 au niveau haut (High), D1 au niveau bas et D0 au niveau haut. Donc, on a 0101 qui valent bien 5 en binaire.

Q10)
On recherche le schéma ci-dessous en reprenant l'empreinte du composant.

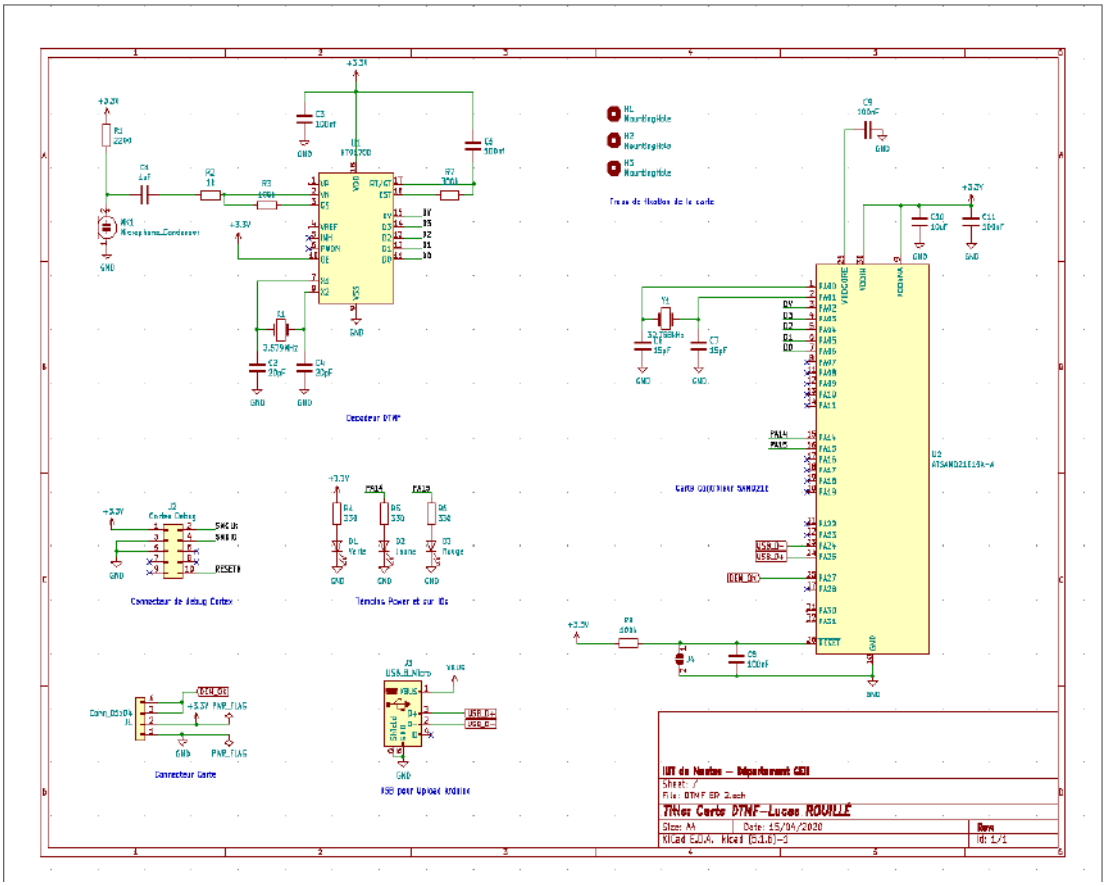


Kicad :

a) La saisie de schéma

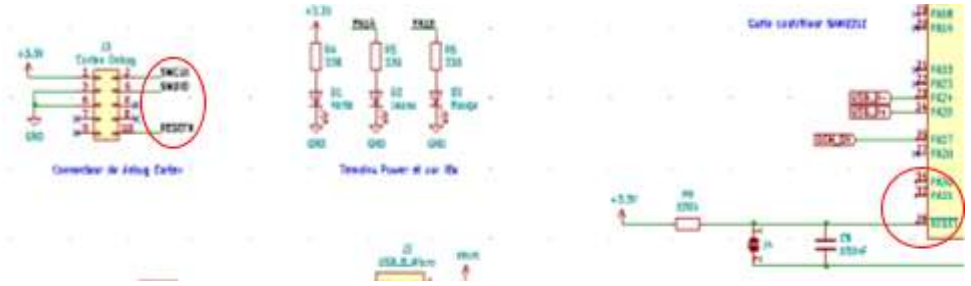
Tout d'abord il faut saisir le schéma sur kicad,

Erreur sur la saisie de schéma :

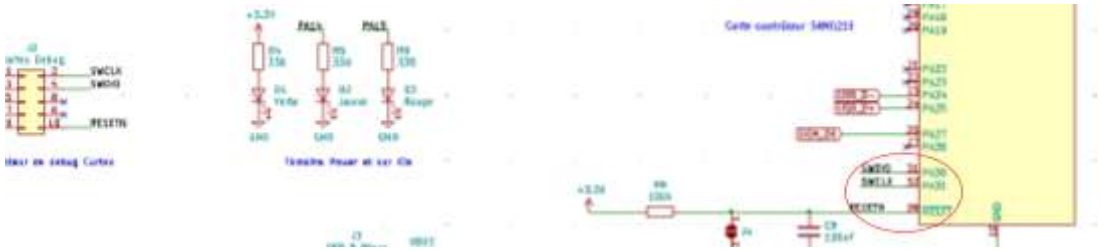


L'erreur n'a pas été reconnue tout de suite, mais lors de la semaine 23 (séance 7).

Si on regarde attentivement la saisie de schéma, on remarque que le cortex de débug n'est pas connecté au SAMD21 sur les pins 31,32 et 26 (SWCLK, SWDIO, RESETN).



Il aurait fallu mettre les noms comme ceci,



b) Les empreintes

Une fois celui-ci terminé, on assigne les empreintes :

1	C1	1uF	Capacitor_SMD:C_0805_2012Metric
2	C2	20pF	Capacitor_SMD:C_0805_2012Metric
3	C3	100nf	Capacitor_SMD:C_0805_2012Metric
4	C4	20pF	Capacitor_SMD:C_0805_2012Metric
5	C5	100nf	Capacitor_SMD:C_0805_2012Metric
6	C6	15pF	Capacitor_SMD:C_0805_2012Metric
7	C7	15pF	Capacitor_SMD:C_0805_2012Metric
8	C8	100nF	Capacitor_SMD:C_0805_2012Metric
9	C9	100nF	Capacitor_SMD:C_0805_2012Metric
10	C10	10uF	Capacitor_SMD:C_0805_2012Metric
11	C11	100nF	Capacitor_SMD:C_0805_2012Metric
12	D1	Verte	LED_SMD:LED_0805_2012Metric
13	D2	Jaune	LED_SMD:LED_0805_2012Metric
14	D3	Rouge	LED_SMD:LED_0805_2012Metric
15	H1	MountingHole	MountingHole:MountingHole_2.7mm_M2.5
16	H2	MountingHole	MountingHole:MountingHole_2.7mm_M2.5
17	H3	MountingHole	MountingHole:MountingHole_2.7mm_M2.5
18	J1	Conn_01x04	Connector_JST:JST_PH_S4B-PH-K_1x04_P2.00mm_Horizontal
19	J2	Cortex Debug	Connector_PinHeader_1.27mm:PinHeader_2x05_P1.27mm_Vertical_SMD
20	J3	USB_B_Micro	Connector_USB:USB_Micro-B_Amphenol_10103594-0001LF_Horizontal
21	J4	SolderJumper_2_Open	Jumper:SolderJumper-2_P1.3mm_Open_Pad1.0x1.5mm
22	MK1	Microphone_Condenser	Capacitor_THT:CP_Radial_D10.0mm_P5.00mm
23	R1	2200	Resistor_SMD:R_0805_2012Metric
24	R2	1k	Resistor_SMD:R_0805_2012Metric
25	R3	100k	Resistor_SMD:R_0805_2012Metric
26	R4	330	Resistor_SMD:R_0805_2012Metric
27	R5	330	Resistor_SMD:R_0805_2012Metric
28	R6	330	Resistor_SMD:R_0805_2012Metric
29	R7	300k	Resistor_SMD:R_0805_2012Metric
30	R8	100k	Resistor_SMD:R_0805_2012Metric
31	U1	HT9170D	Package_SO:SOIC-18W_7.5x11.6mm_P1.27mm
32	U2	ATSAMD21E15A-A	Package_QFP:TQFP-32_7x7mm_P0.8mm
33	X1	3.579MHz	Crystal:Crystal_SMD_HC49-SD
34	Y1	32.768kHz	Crystal:Crystal_C26-LF_D2.1mm_L6.5mm_Vertical

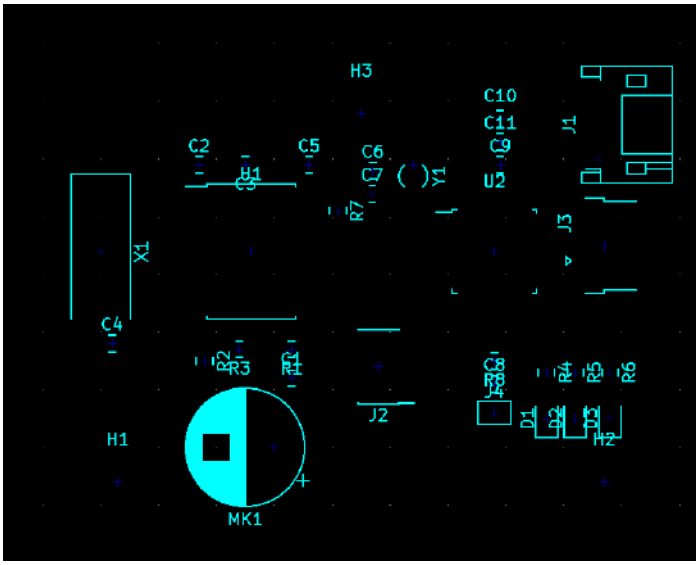
Erreur sur les empreintes :

Il n'y a pas eu d'erreur sur les empreintes.

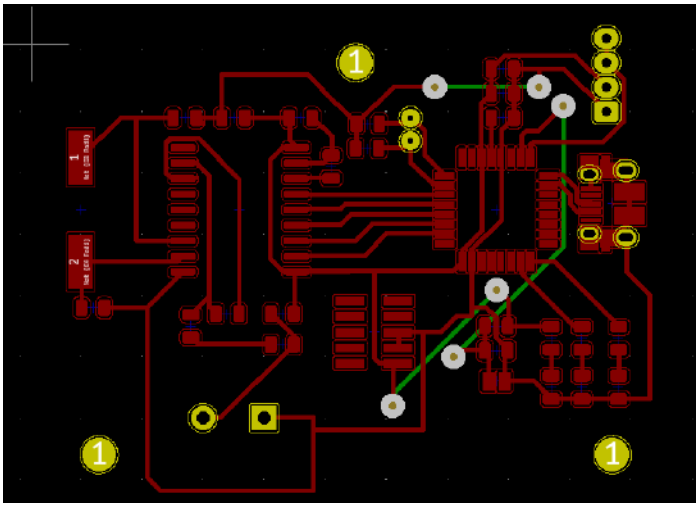
On peut donc désormais passer au routage, pour un souci de facilité, on va reprendre les indications données au projet d'ER 1.

c) Routage

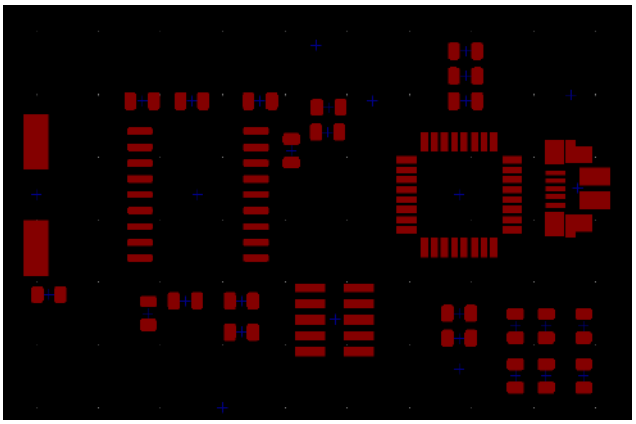
Couche F Skill



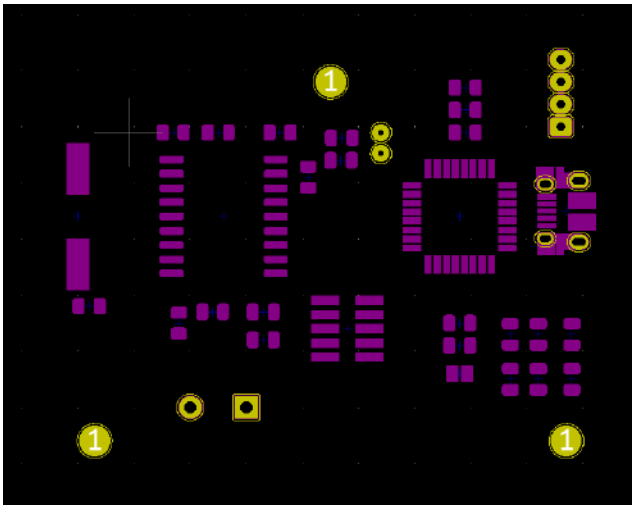
Couche FCu et BCu :



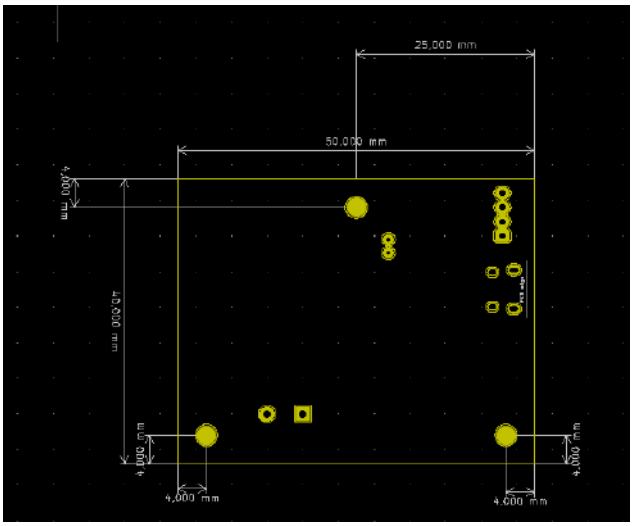
Couche F.Paste :



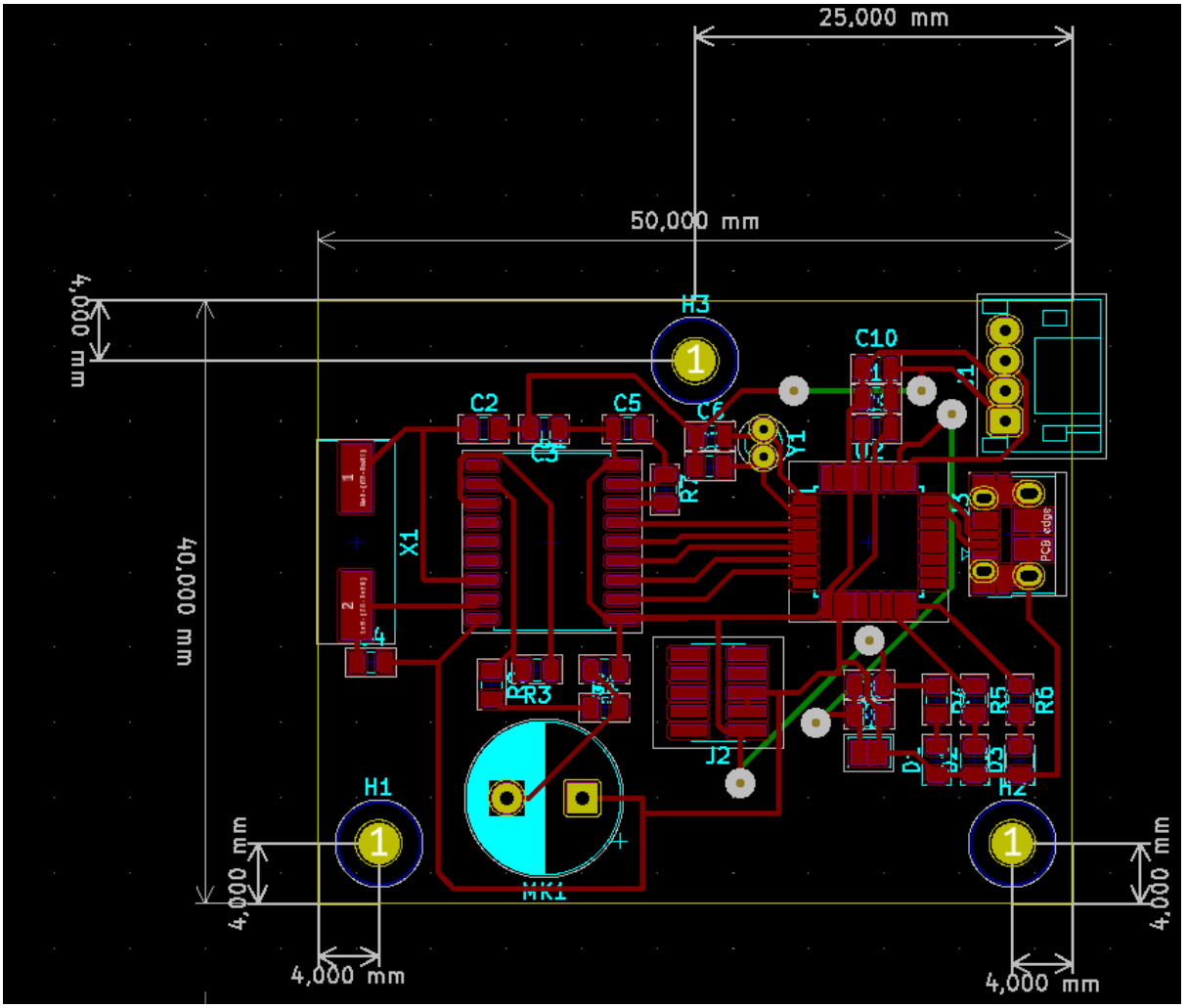
F.Mask et B.Mask :



Perçage et de dimension :

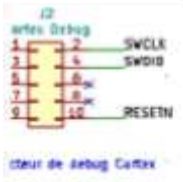


Vue d'ensemble :

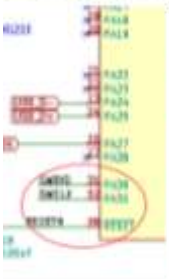


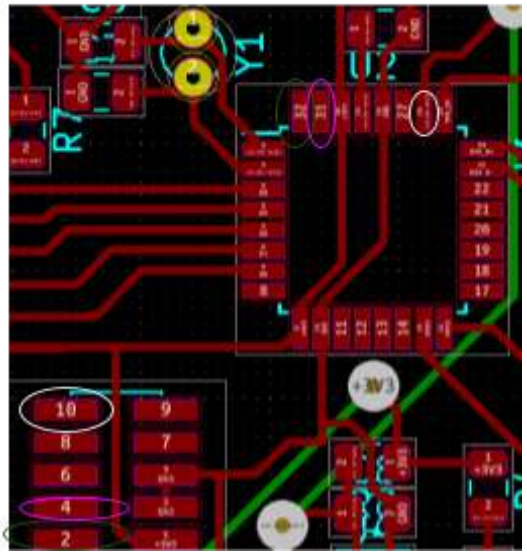
Erreur sur le routage :

Lors de ce routage, j'ai rencontré quelques difficultés telles que des problèmes d'empreintes, de taille, de piste et j'ai dû le refaire 5 à 6 fois afin d'avoir ce rendu (très aligné et en cohérence dans les emplacements des composants). Nous avons ensuite envoyé la carte pour la faire imprimer. Cependant, comme expliqué dans la partie schéma, j'ai oublié de connecter les pins (SWCLK, SWDIO, RESETN).



Les pins du cortex à connecter sont les pins : 2,4 et 10 ce qui représente pour le SAMD21 les pins: 32,31 et 26

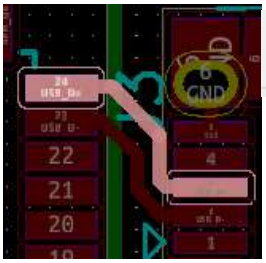




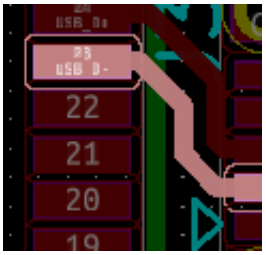
2e problème, dans le routage (repéré séance 9 semaine 24)

Lors du boot loader, la carte était initialisée, mais ne se faisait pas reconnaître par un PC. Ceci est dû à un écart relatif trop important de longueur entre D+ et D- soit

$$\Delta écart\ relatif\ = \frac{|Valeur\ longueur\ D+ - Valeur\ longueur\ D-|}{Valeur\ longueur\ D+} * 100 = \frac{5.057-4.865}{5.057} * 100 = 3.8\%$$



Longueur 5,057 mm

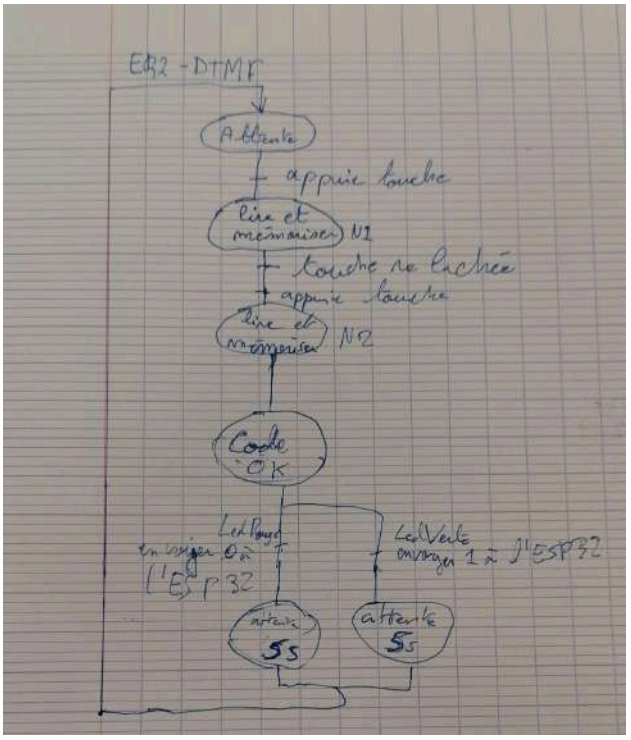


Longueur 4,865 mm

Cet écart est trop élevé et nuira par la suite au bon fonctionnement du port USB.

Tinkercad :

Avant l'impression de la carte, j'ai commencé à faire des essais sur Tinkercad.
Tout d'abord, on schématise le fonctionnement du DTMF :



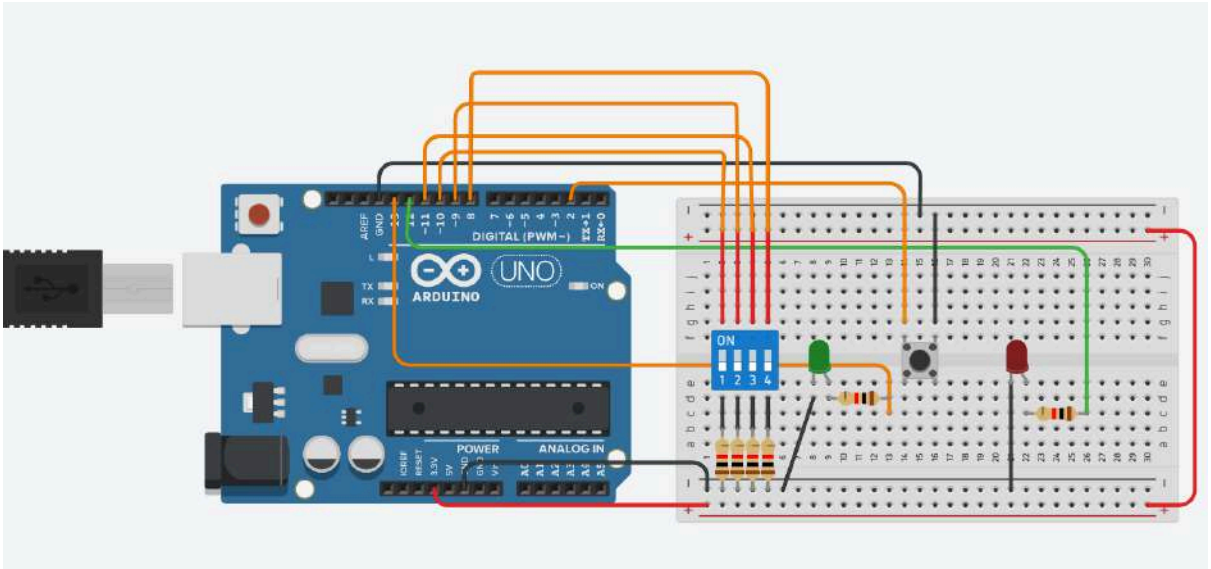
On explique ici de manière synthétique comment on utilise un DTMF sur ce graphe.

Tout d'abord, on appuie sur une touche DTMF depuis une application sur téléphone. La carte va mémoriser le chiffre sur lequel on a appuyé grâce à la fréquence du son émis (lire et mémoriser). Ensuite, une fois que la touche est relâchée, on peut de nouveau répéter l'opération. Enfin, si le code est bon, la LED verte s'allume durant 5 secondes et l'information est donnée à l'ESP32 sinon, la Led est rouge et on devra atteindre 5 secondes qu'elle s'éteigne pour recommencer la saisie.

Tinkercad :

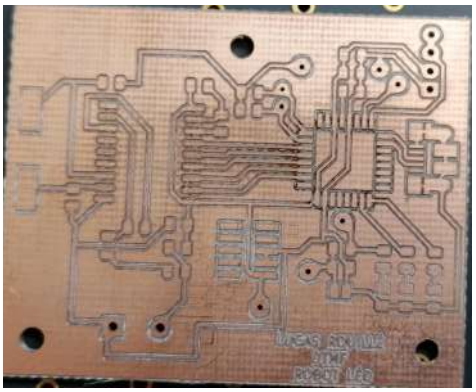
Une fois le routage fini, je me suis dirigé sur Tinkercad afin de pouvoir démarrer la programmation et de comprendre le fonctionnement du système DTMF.

On observe ici la carte, on a un Switch représentatif d'une touche, le bouton poussoir aura le rôle du DTMF, c'est-à-dire mémoriser et vérifier la touche. On obtient le circuit ci-dessous :



La carte :

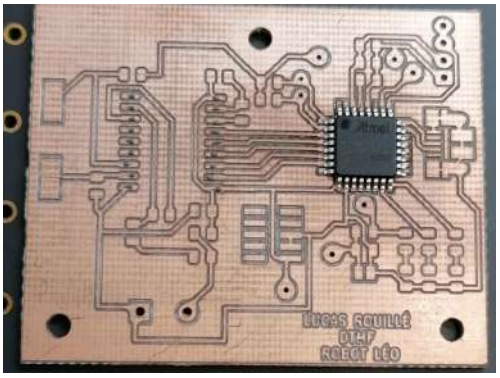
Carte vierge :



Nous observons ici la carte quand elle a été imprimée.
On remarque que les trous de l'USB n'ont pas été faits.

Par la suite, on devra donc percer la carte.

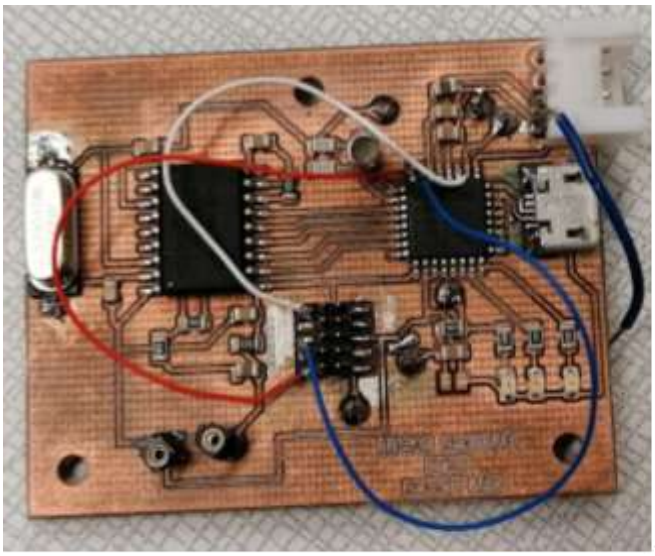
Carte pendant le dépôt de la pâte à braser :



Nous observons ici la pâte à braser en train d'être déposée.

On se sert des machines comme en ER1 pour réaliser le placement PCB.

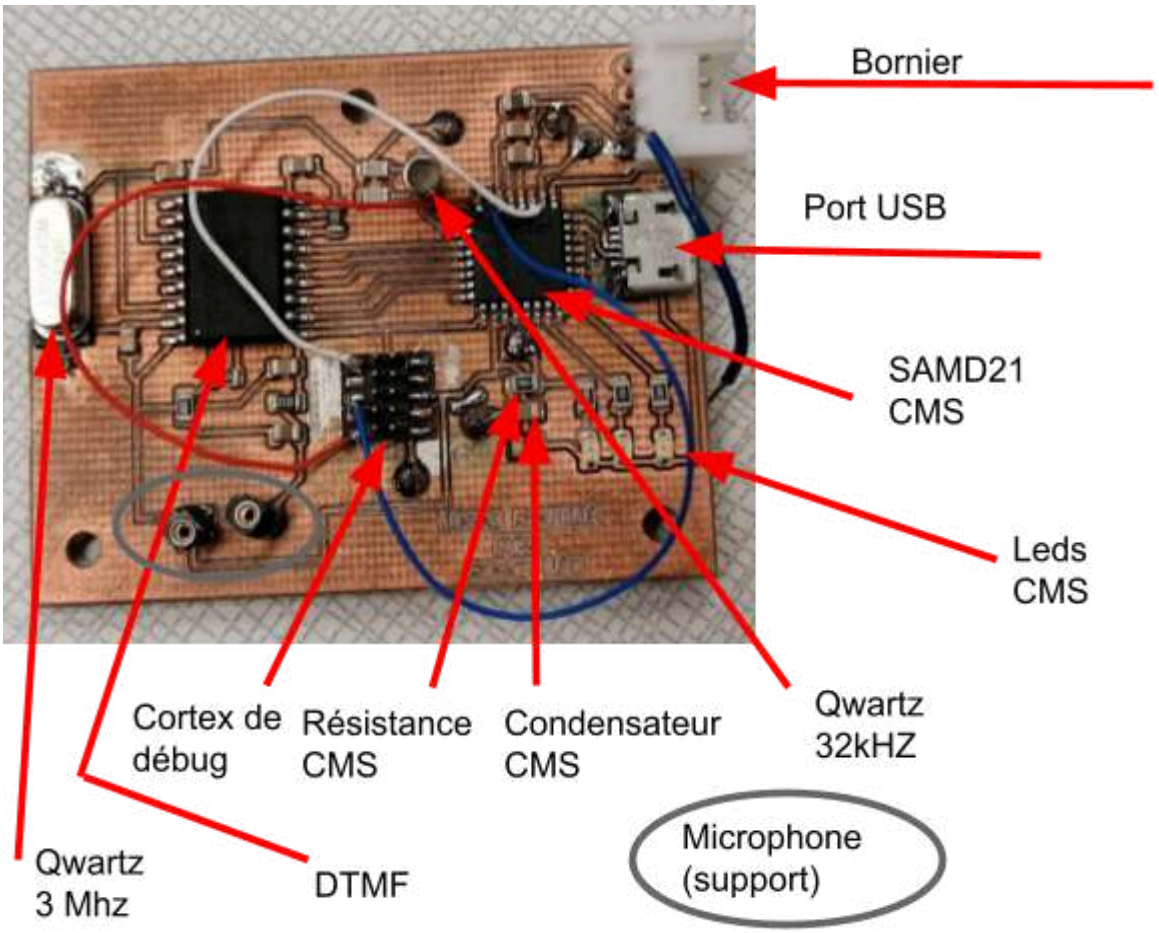
Soudures et carte complète :

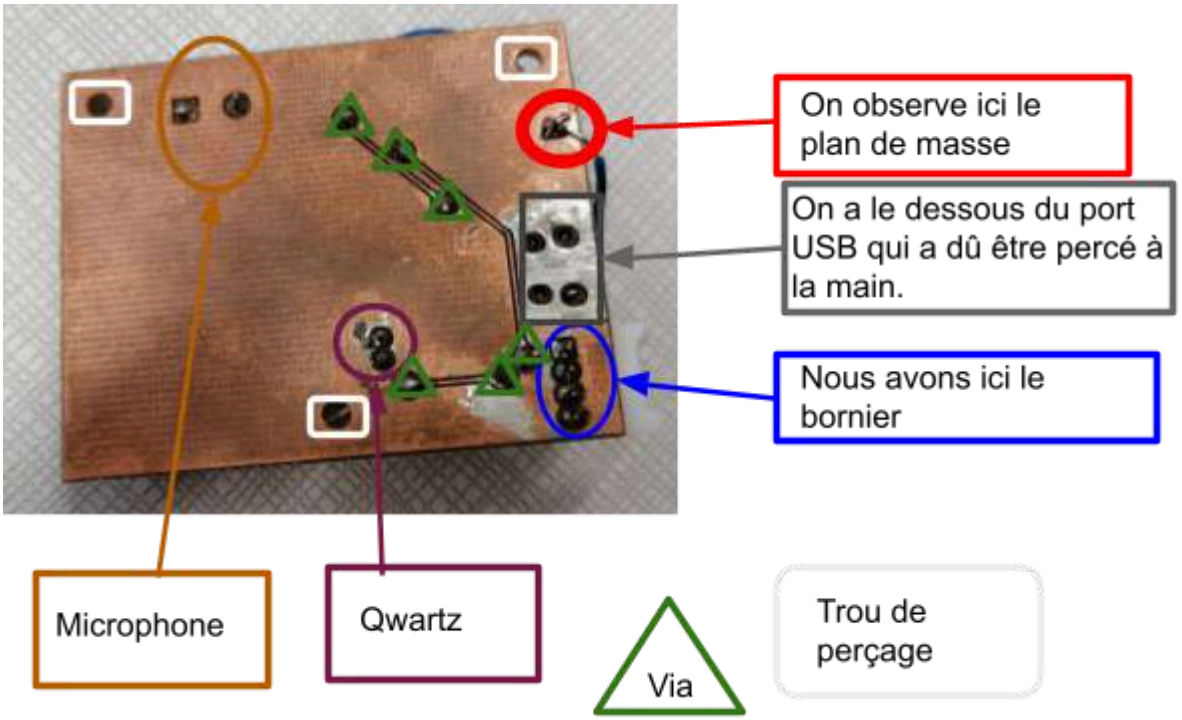


Tout a été soudé par moi et mon équipe (sans l'aide d'un professeur).
On remarque 4 fils ces fils représentent les erreurs précédentes, (SWCLK,SWDIO, RESETN)

Le dernier est relié au GND afin de faire un plan masse afin de pouvoir faire les tests avec plus de facilité

Tout les test de continuité ont été effectué il y a eu aucun court-circuit cependant les leds ont été déposé sans faire attention à la polarisation de celle-ci.

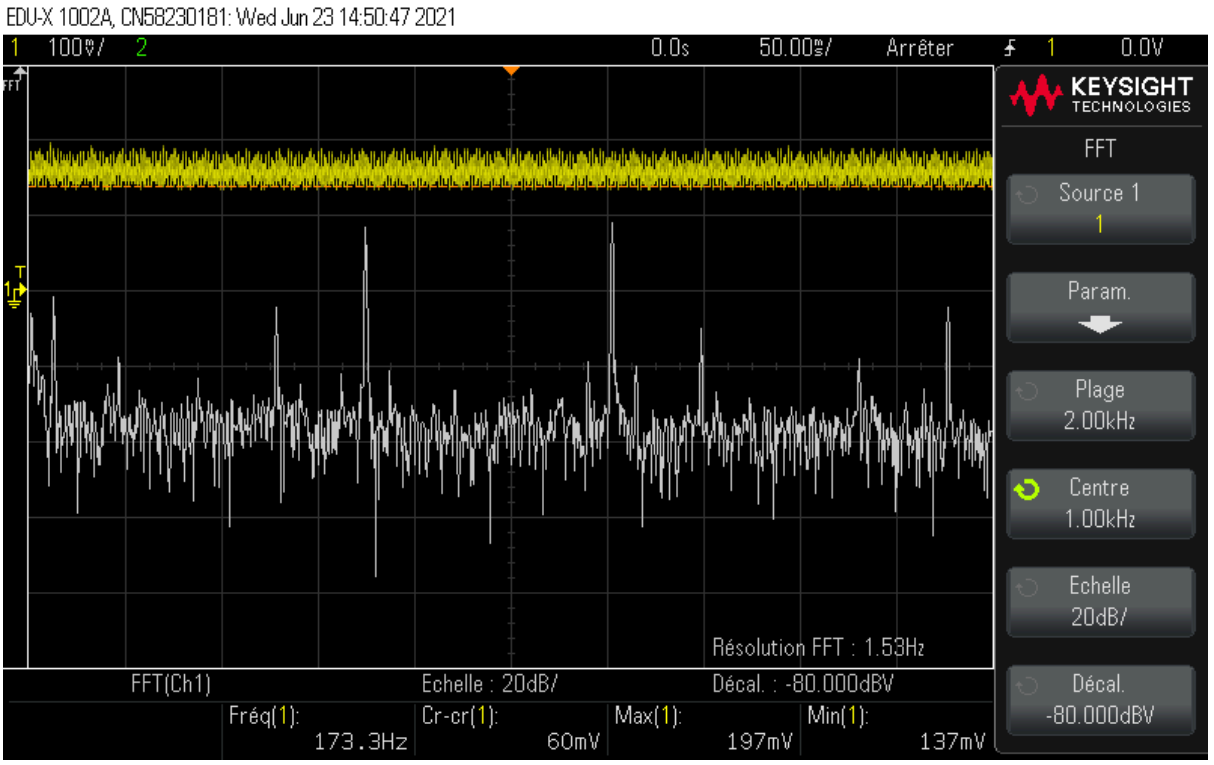

Bornier
Port USB
SAMD21
CMS
Leds
CMS
Qwartz
32kHZ
Condensateur
CMS
Résistance
CMS
Cortex de
débug
Qwartz
3 Mhz
DTMF
Microphone
(support)


On observe ici le
plan de masse
On a le dessous du port
USB qui a dû être percé à
la main.
Nous avons ici le
bornier
Microphone
Qwartz
Via
Trou de
perçage

Test

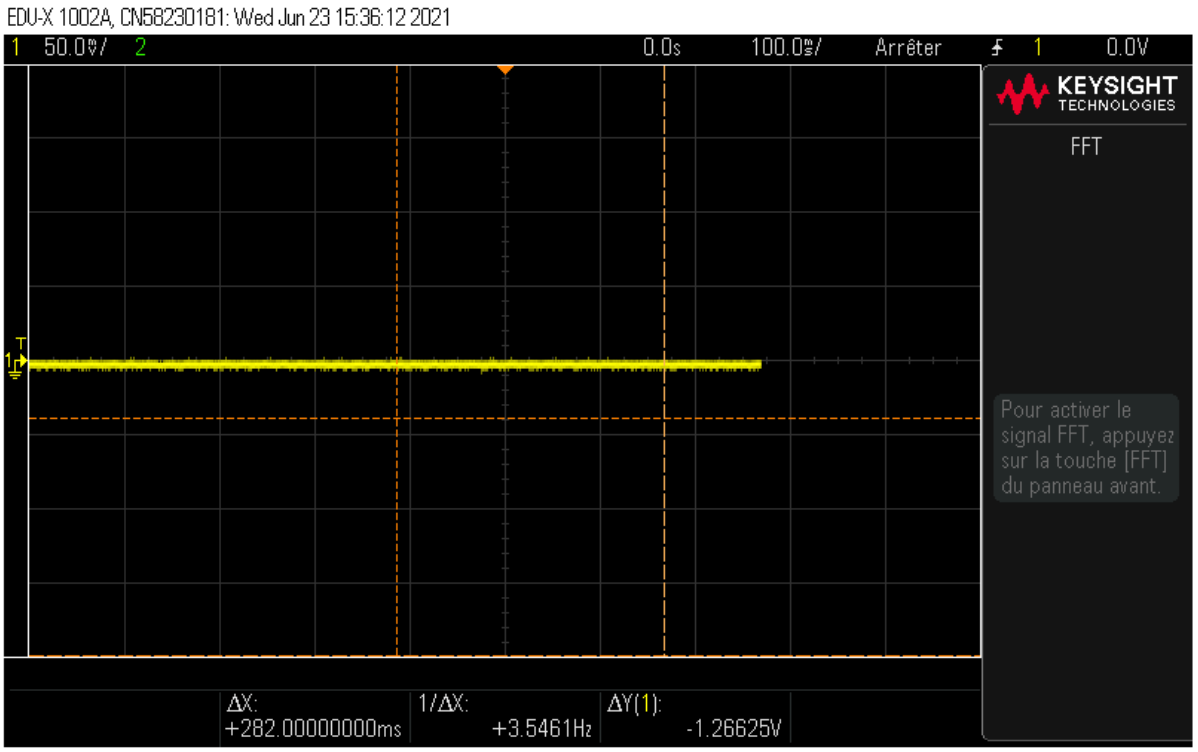
On met le microphone sur le support, on alimente la carte en 3.3 V continue et on observe le signal sur un oscilloscope en mode AC et FFT comme ça, à chaque fois que la carte détecte un signal, on observera un pic de son.

On observe ici le signal qui rentre dans la pin 3 du DTMF lorsque l'on émet un son.



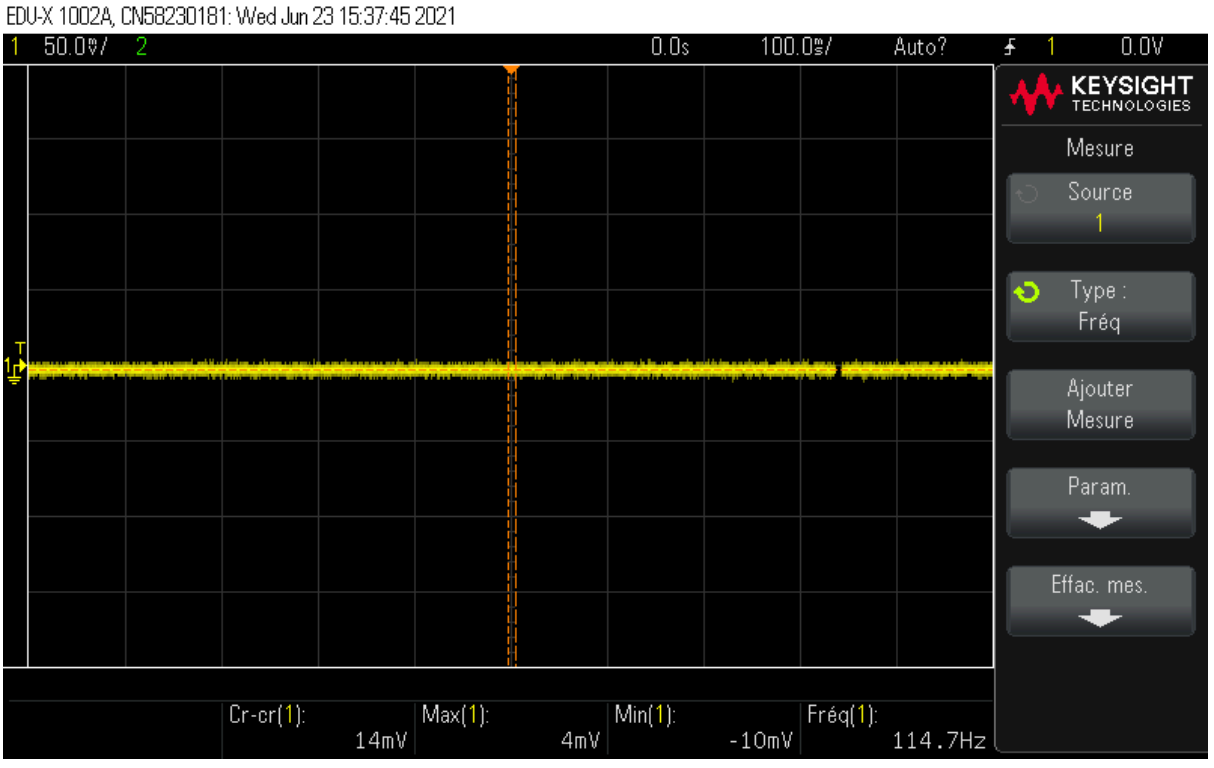
On remarque que des pics en dB apparaissent.

On observe désormais l'état du Quartz de 3.57 MHz :



On remarque que l'on n'a malheureusement rien en sortie.

Maintenant, c'est au tour de DV. On effectue la même manipulation.



Mais comme on peut l'observer, on n'a rien non plus sur la pin DV

Le Programme :

```
#define D0 6
#define D1 5
#define D2 4
#define D3 3
#define DV 2
#define LEDJAUNE 14

#define Dem_OK 27

int Di0 = 0;
int Di1 = 0;
int Di2 = 0;
int Di3 = 0;
int code[4] = { 0, 0, 0, 0 };
int codebon[4] = { 1, 2, 1, 2 };
int securite = 0;
int i = 0;
int Digits;

void setup() {
 // Communication avec le pc
 Serial.begin(9600);
 // Sorties
 pinMode(14, OUTPUT);
 pinMode(15, OUTPUT);
 pinMode(Dem_OK, OUTPUT);
 digitalWrite(Dem_OK, 1);

 // Entrées
 pinMode(D0, INPUT);
 pinMode(D1, INPUT);
 pinMode(D2, INPUT);
 pinMode(D3, INPUT);
 pinMode(DV, INPUT);
}

void loop() {
```

```
do {
  // Lecture de DV
  if (digitalRead(DV)) {
    // Lecture des 4 bits du  DTMF
    Di0 = digitalRead(D0);
    Di1 = digitalRead(D1);
    Di2 = digitalRead(D2);
    Di3 = digitalRead(D3);

    // Assemblage des bits pour trouver la touche appuyée
    Digits = Di0 + (Di1 << 1) + (Di2 << 2) + (Di3 << 3);

    // Ajout de la touche appuyée dans le code (ième chiffre)
    code[i] = Digits;

    // Confirmation d'enregistrement avec allumage de la LED
    digitalWrite(14, 1);
    delay(500);
    digitalWrite(14, 0);

    i++;
  }
// Lecture de 4 chiffres
} while (i < 4);

// RAZ i
i = 0;

// Vérification du nombre de chiffres valides
for (int i = 0; i < 4; i++) {
  if (code[i] == codebon[i]) {
    securite++;
  }
}

// Si 4 chiffres corrects, code correct
if (securite > 3) {
  Serial.println("Code Bon");

  // Pulsation à 0 sur DEM_OK
  digitalWrite(Dem_OK, 1);
  delay(500);
  digitalWrite(Dem_OK, 0);
```

```
    // Allumage de la LED de confirmation
    digitalWrite(15, 1);
    delay(500);
    digitalWrite(15, 0);
  // Sinon, code érroné
  } else {
    Serial.println("CodeFaux");

    // Clignotage de la LED 4 fois pour signifier l'erreur
    for (int j = 0; j < 3; j++) {
      digitalWrite(15, 1);
      delay(500);
      digitalWrite(15, 0);
      delay(500);
    }
  }

  // RAZ de securité pour retenter un code
  securite = 0;
}
```

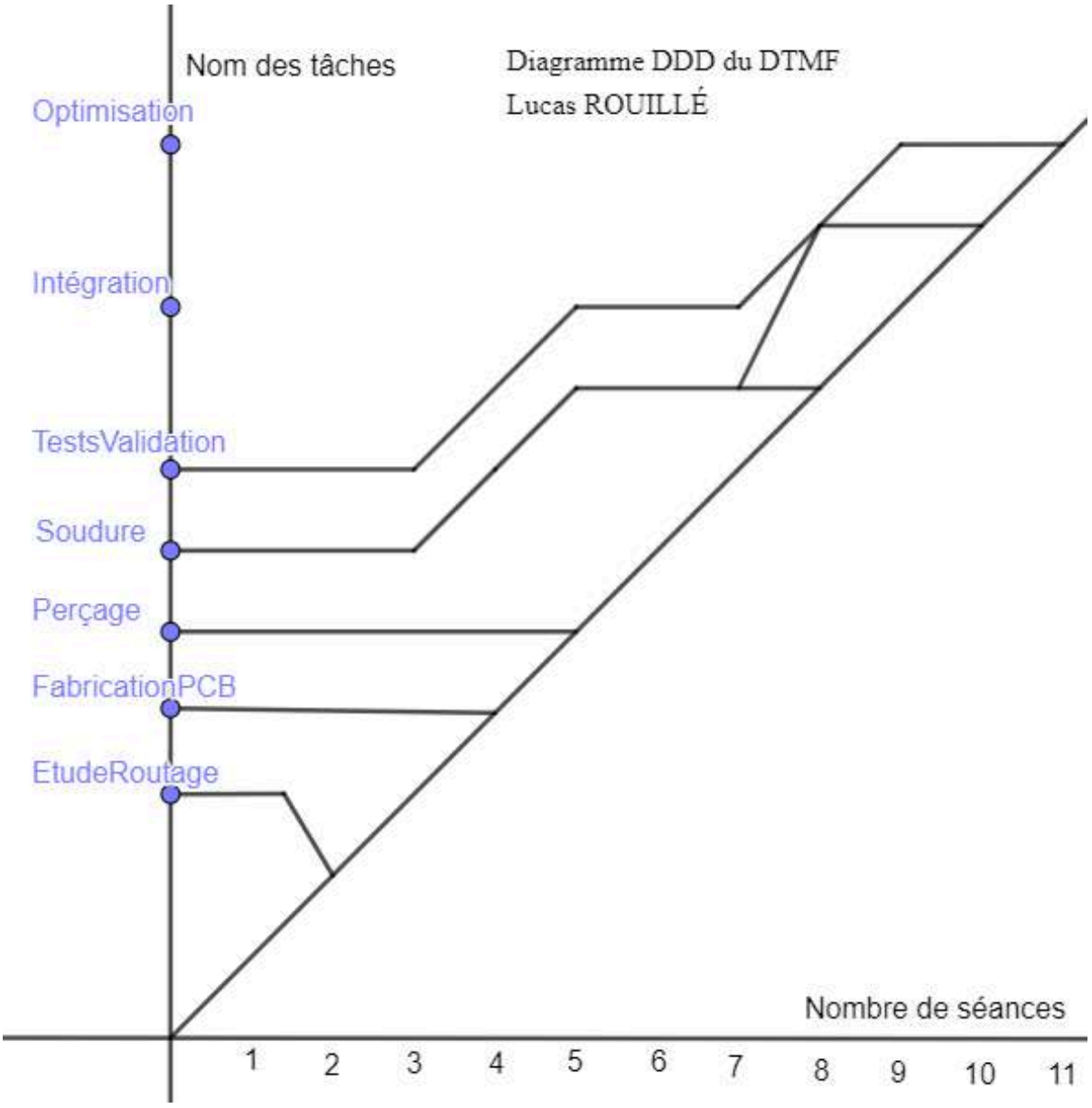
Le code a été réalisé durant la semaine 7 à la fin de celle-ci, j'ai voulu le tester et donc je suis allé voir un professeur afin de faire le boot loader sur ma carte cependant, c'est comme ça que je me suis rendu compte de mes erreurs par rapport au cortex comme expliqué précédemment. Suite au problème rencontré avec l'USB, je n'ai malheureusement pas pu tester ce code sur ma carte.

L'emploi du temps :

Séance	Groupes A/B	Groupes C/D	Objectifs
1	15	16	Etude & Routage
2	15	16	Etude & Routage
3	vacances	vacances	Routage
4	21	20	Fabrication des PCB
5	21	20	Perçage/Soudures
6	23	22	Soudures / Tests & Validations
7	23	22	Tests & Validations
8	24	23	Tests & Validations
9	24	23	Intégration au robot
10	25	24	Intégration au robot
11	25	24	Optimisation
coupe			

On a ici l'emploi du temps, il a été très difficile de le respecter dans la fonction DTMF aucune personne de notre classe de TD a réussi à correctement finir la fonction (n'est allé jusqu'à l'étape d'intégration).

J'ai regroupé ici toute l'évolution du projet de ma partie dans ce diagramme



On remarque que dès le début, j'ai très bien commencé, car j'étais en avance par rapport à l'emploi du temps, cependant, il était prévu de continuer pendant les vacances et donc cette avance n'a eu aucune utilité, parce que je ne pouvais pas passer aux tâches suivantes.

Ensuite, tout s'est bien passé jusqu'à la soudure sur laquelle il manquait des composants pour les premières séances de soudures, ceci a eu comme incidence de me contraindre de me concentrer sur d'autres aspects du projet tel que la programmation, donc j'ai commencé à me servir de Tinkercad. Une fois les soudures "finies".

Je suis passé aux tests de validations, c'est malheureusement seulement à partir de là que je me suis rendu compte de la plupart de mes erreurs tout d'abord il fallait tester si toutes les continuités étaient bonnes ce qui était le cas j'ai eu aucun court-circuit et lors de l'alimentation pas de problème vis-à-vis de mon 3.3V et de mon GND mais c'est lors de mon bootloader qui n'a pas fonctionné que j'ai eu mes premiers problèmes j'ai de nouveau soudé ce qui explique le pic sur le DDD mais comme ce problème a été découvert en toute fin de séance ainsi, la tâche soudure et la tâche TestsValidations a pris du retard en même temps.

Par la suite, j'ai dû repousser les tests de validation et continuer sur la soudure en soudant trois fils comme expliqué précédemment et j'en ai profité pour faire un plan de masse. Ainsi, j'ai pu finir définitivement la tâche de soudure et repasser aux tests.

Enfin, on arrive à la dernière séance, il faut donc finir au plus vite les tests, mais malheureusement, même si le micro marche bien, il n'y a aucun signal pour DV et aux bornes du Qwartz.

Je n'aurais malheureusement pas pu intégrer ma carte au robot et je n'aurais également pas pu optimiser ma carte.

Conclusion DTMF :

J'ai été très satisfait du module ER2, j'ai trouvé le projet très intéressant et enrichissant. Contrairement au module ER1, l'encadrement était quasiment inexistant et j'ai trouvé ceci très enrichissant, même si ce n'est pas un projet en entreprise, je l'ai vécu comme tel. On a travaillé en groupe et ceci a mené à une pression. J'ai vécu cette pression de manière positive, car ça m'a poussé à faire attention à ne pas faire de court-circuit, même si, malheureusement, j'ai eu quelques problèmes avec la carte. J'ai par ailleurs appris ce qu'était un DTMF et comment ça fonctionne. J'ai aussi beaucoup apprécié la liaison entre le module CP2 et ER2 ayant beaucoup apprécié la matière CP2, j'ai également beaucoup apprécié la matière ER2. Je suis quand même déçu de ne pas avoir pu finir ma partie, mais le projet m'aura permis de combler certaines lacunes comme le routage, les soudures.

Budget final :

Malheureusement, le projet n'a pas pu être fini en 11 semaines, cependant, on notera qu'on n'a pas dépassé le budget des composants, soit 60.63 €.

Par contre, si on devait finir complètement le projet, il faudrait encore environ 3 séances, donc le budget main d'œuvre augmentera. Pour rappel, nous avions 11 séances, ce qui représente 660 €.

Soit 13.33*4*3 = 160 €, soit un coût de main-d'œuvre total de 660+160 = 820 € et un coût total de 820+60.63= 880.63€.

On en conclut que le budget prévu à l'effigie du projet est respecté, car on avait un budget total de projet de 1008.9 €.

On tient à rappeler que les coûts liés à la location des locaux, du matériel, l'électricité [...] ne sont pas comptés.

Tableau de bord :

Chef de projet : Lucas ROUILLÉ

	FP1 : Balise Dimitri	FP2 : Gestion du Robot-Theodros	FP4 : Sonar Pierre	FP7 : DTMF Lucas
Séance 1	Début			
Séance 2	Début routage	Programmation de base (clignotement des Led)	Début routage	Début du routage, schéma fini
Séance 3	Vacances			
Séance 4	Fin routage et début fabrication	Fabrications des PCB et soudure	fin routage, démarrage partie théorique	Fin routage en attente de validation
Séance 5	Fin cms et début soudure	Fabrication laser du châssis	commencement Cms et début soudure	CMS fini-début soudure
Séance 6	Fin soudure et début des tests	Programmation des moteurs	Fin CMS, finir la soudure	Continu soudure (trouver les composants)
Séance 7	Continu les tests et début programmation	Test de la programmation	Fin soudure début phase de tests et programmation	Début Compte rendu + programmation + premier tests
Séance 8	Programmation	Programmation	Continu les tests et programmation	tests avancé + problème avec la carte retour soudure pour les pins du cortex
Séance 9	Programmation continu test	Programmation et aide aux autres groupes (soudures etc..)	Continu les tests fin programmation	Soudure fini, test pour le programme
Séance 10	Problème programmation débugage	Programmation et aide aux autres groupes(soudures etc..)	Fin tests	Problème USB on continu les test + partie théorique
Séance 11	Fin programmation	Programmation et intégration du sonar	Intégration au robot	Compte rendu et partie théorique + dernier test

Conclusion du projet :

Pour conclure, nous n'avons pas pu finir le robot Léo en 11 séances, malheureusement certaines parties demandaient plus de travail que d'autres, par exemple la FP2 est très dépendante des autres et donc l'avancement était assez lent, cependant, on a tous apprécié ce module. La dynamique de groupe était très bonne, il n'y a eu aucun conflit et il y a eu beaucoup d'entraide dans le groupe. On a su surmonter la plupart des problèmes que l'on a pu rencontrer et nous avons eu une grande communication, à chaque problème, nous allions informer le reste du groupe, nous avons organisé une réunion ad-hoc durant le projet séance 8 (milieu fin de projet) afin de pouvoir comprendre où se situe chaque membre du groupe. On a tous appris de nos erreurs et nous sommes satisfaits de ce projet.