Systèmes de télécommunications pour applications embarquées

BOULEGHALEGH Aya

BARKOUM Betra Félix

Année universitaire

2021/2022

Contents

1 Introduction

La modulation d'amplitude en quadrature (QAM) est utilisée dans plusieurs systèmes de transmission de donnée tels que wifi, téléphone portable etc. QAM est une méthode de combinaison de deux signaux de modulation d'amplitude (AM) en un seul canal.QAM est utilisé pour atteindre des niveaux élevés d'efficacité d'utilisation du spectre. Ceci est accompli en utilisant à la fois les composantes d'amplitude et de phase pour fournir une forme de modulation. Dans ce scénario, le signal QAM est livré avec deux porteuses. Chacun a la même fréquence mais diffère en phases de 90 degrés, soit un quart de cycle, qui est à la base du terme quadrature. Dans la suite de ce projet, nous allons réaliser en premier temps sur LTSPICE, un émetteur RF illustré sur la figure 1, et par la suite nous allons piloter un synthétiseur de fréquence analogique (PLL ADF4360-7) avec un module Arduino UNO à l'aide d'un protocole SPI (Serial Peripheral Interface).

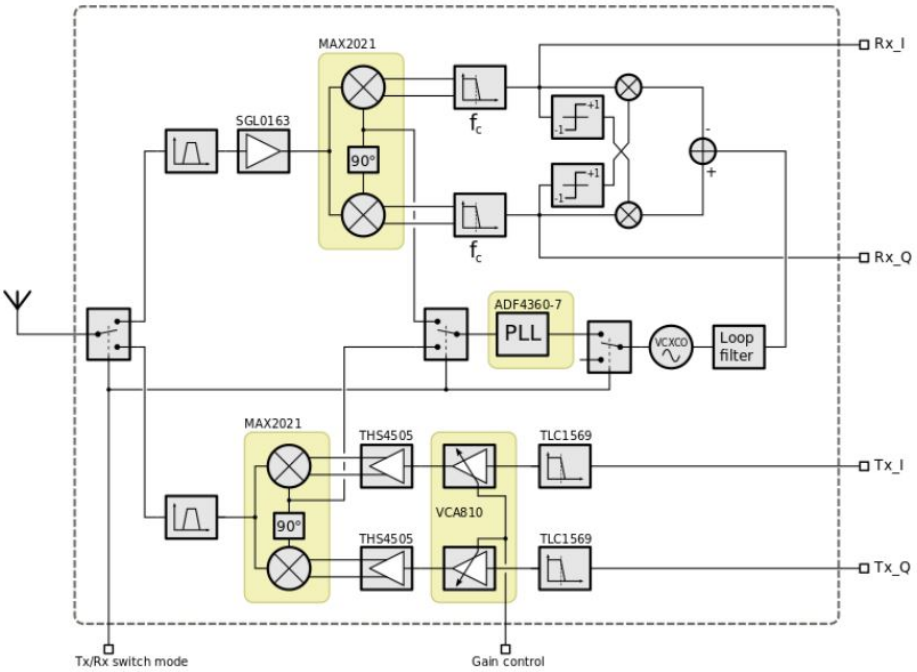
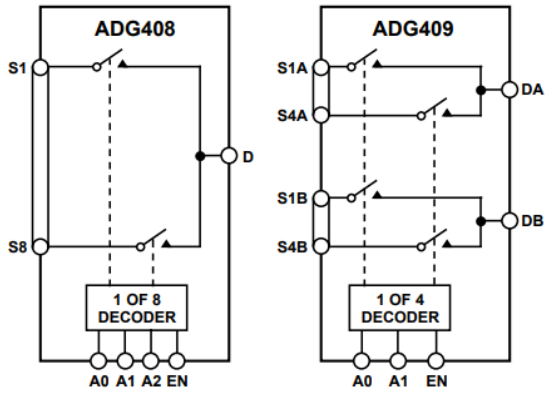


Figure 1: schéma simplifié du système RF complet

2 Conception de l'émetteur QPSK et 16-QAM

2.1 Convertisseur numérique-analogique

Afin de faciliter la réalisation de ce projet, les sources NRZ-2 et NRZ-4 ont été fourni par l'enseignant. Ces sources génèrent des bits de manière aléatoire. Ces bits sont ensuite passés à travers un convertisseur numérique analogique (DAC). Dans notre projet, nous avons utilisé un multiplexeur (ADG408) pour réaliser la fonction du DAC. En fait, le ADG408 est un multiplexeur analogique de faible puissance de consommation, pouvant fonctionner sous tension symétrique. Après avoir consulter la fiche technique du ADG408, nous avons constaté que ce multiplexeur peut être utilisé dans des systèmes de télécommunication. Ce multiplexeur analogique comporte 8 entrées, trois adresses pour adresser les 8 différentes entrées en sortie. Le diagramme fonctionnel ainsi que la table de vérité du ADS408 est illustrés sur la figure 2.



ADG408 Truth Table

A2	A1	A0	EN	ON SWITCH
X	X	X	0	NONE
0	0	0	1	1
0	0	1	1	2
0	1	0	1	3
0	1	1	1	4
1	0	0	1	5
1	0	1	1	6
1	1	0	1	7
1	1	1	1	8

Figure 2: Diagramme fonctionnel et table de vérité du ADG408.

Comme nous pouvons constater, notre multiplexeur comporte huit entrées analogiques, et trois adresses. Nous allons choisir quatre entrées (niveau de tension arbitrairement choisis tout en respectant les écarts entre les niveaux de tensions) et les entrées non utilisées seront connectées à la masse. Nous avons besoin de deux adresses (I0 et I1) pour la voie I et Q0, Q1 pour la voie Q, du coup le A2 sera lié à la masse. Le schéma de montage réalisé sur LTSPICE est illustré sur la figure 3.

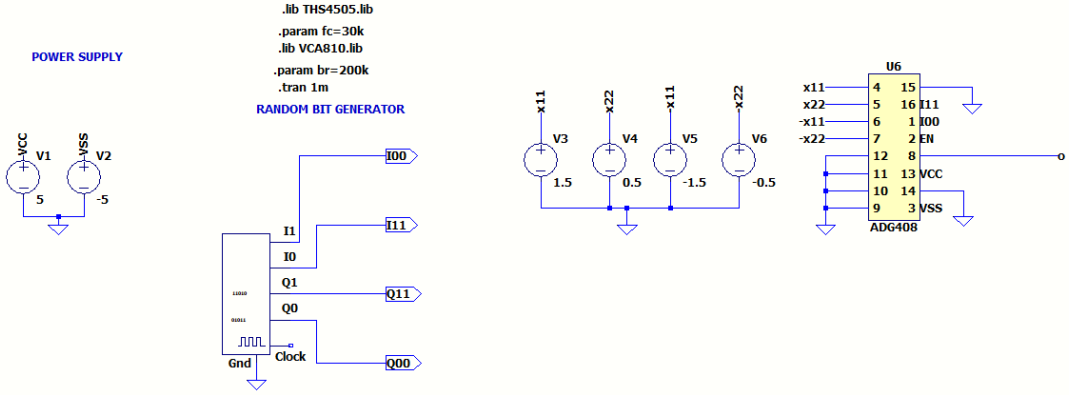


Figure 3: Montage sur LTSPISE

En consultant la fiche technique du multiplexeur analogique, nous avons réalisé le montage sur LTSPICE en respectant les broches du composant comme illustré en figure 4.
Le résultat de la simulation illustrée sur la figure 5 montre effectivement un niveau de quatre états (1.5v,0.5v, -0.5v et 1.5v) arbitrairement choisis en sortie du multiplexeur.

L'avantage de cette technique par rapport aux autres DAC est que nous pouvons directement choisir les tensions (positive et négative) que nous aurions en sortie. Nous pouvons directement avoir une tension négative en sortie sans toutefois besoin d'utiliser un soustracteur en sortie du multiplexeur. Ceci permet de limiter le courant consommé par le système et gagner en temps. Pour la suite de notre projet, nous avons choisi des niveaux de tensions suivants : **10 = +v = 2mv, 01 = -v = -2mv, 00 = -3v = -6mv, 11 = +3v = 6mv**.

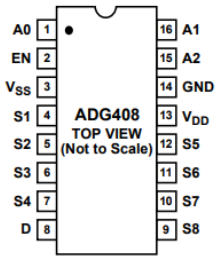


Figure 4: Brochage du ADG408

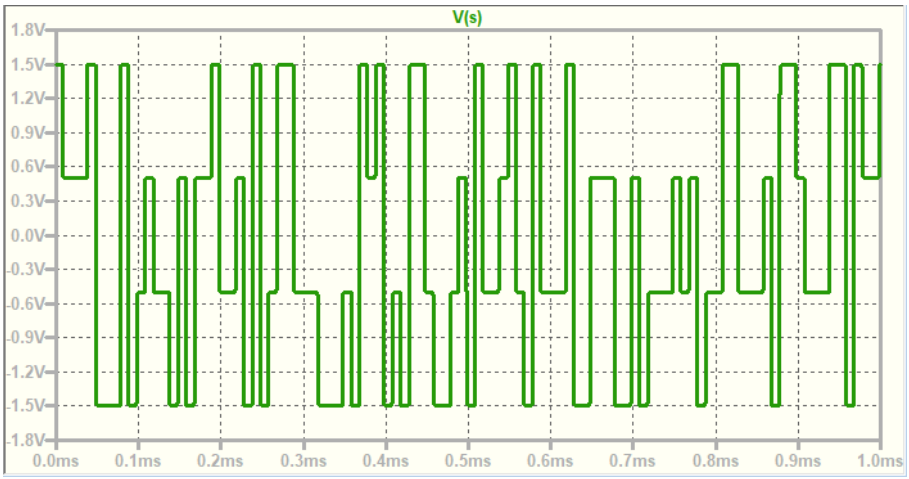


Figure 5: Signal de sortie du multiplexeur analogique.

2.2 Filtrage passe-bas du signal en bande de base

Les signaux provenant des DAC des voies I et Q sont filtrés avec un filtre passe bas d'ordre 10. Ce filtrage permet d'éliminer les signaux qui viennent s'ajouter au signal utile que nous souhaitons moduler. Ce filtrage est réalisé par le composant LTC1569. Ce filtre a une résistance d'entrée très grande de l'ordre de 10M et une résistance de sortie très faible (5 Ohm). Du coup nous n'avons pas besoin de faire une adaptation d'impédance. Le filtre à la sortie du DAC de la figure 6 limite la bande passante des niveaux de tension du DAC. A partir des diagrammes de l'œil, on n'a choisi une fréquence de coupure de fc = 30kHz. Comme le montre le diagramme de l'œil de la figure 8, les yeux sont ouverts dans le cas d'une fréquence de coupure de 30kHz, par contre pour une fréquences inferieure a 30kHz, les yeux sont fermés. La figure 7 represente l'entrée et la sortie du filtre pour le cas de quatre niveaux de tension (1.5v,0.5v, -0.5v et 1.5v).

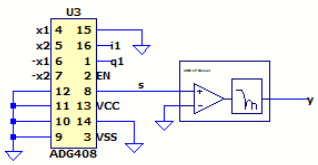


Figure 6: Filtre relié à la sortie du DAC.

Avant de moduler le signal à transmettre, il est important d'augmenter la puissance du signal. Ceci est réalliser par un amplificateur à gain variable.

Figure 7: Niveaux de tensions filtrées.

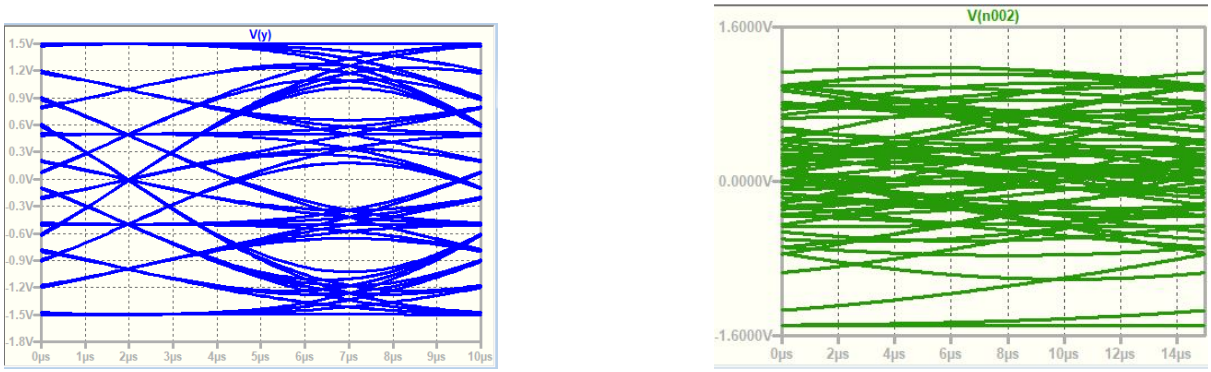


Figure 8: Diagramme de l'œil pour fc = 30kHz et fc < 30KHz.

2.3 Amplification à gain variable

Le VCA810 est un amplificateur à gain variable. A partir d'une tension contrôlable VC on peut obtenir un gain allant de -40dB à 40dB. D'après la fiche technique de ce composant, il possède un faible bruit de l'ordre de $(2.4\text{nV})/\sqrt{Hz}$, une bande passante à gain contrôlable allant jusqu'à 25MHz , un courant en sortie très élevé de l'ordre de 60ma et en fin un courant d'alimentation raisonnable de 24.8ma. En regardant tous ces avantages par rapport aux autres types d'amplificateur à gain variable(AD8338), nous allons utiliser cet amplificateur dans notre projet. Le tableau suivant illustre la comparaison entre le VCA810 et l'AD8338.

Table 1: Comparaison entre le VCA810 et AD8338

	VCA810	**AD8338**
High Gain Adjust Range	+/-40dB	+/-40dB
Low Input Noise Voltage	$(2.4\text{nV})/\sqrt{Hz}$	$(4.5\text{nV})/\sqrt{Hz}$
Gain Control Bandwidth	25MHz	18MHz

A partir du tableau 1, le VCA810 est un meilleur choix pour notre projet. Pour avoir un gain maximal de 40dB, nous avons fixé les quatre niveaux de tension en entrée du multiplexeur analogique respectivement à **6, 2, -2 et − 6mv**. Ce qui respecte la fiche technique du VCA810 qui exige une tension en entrée de l'amplificateur inferieur à **3Vpp** (illustré sur figure 10). On obtient alors une tension **Vpp = 6-(-6) = 12mv** qui est inferieure a 3Vpp. La prochaine étape consiste à chercher la tension VC qui

nous permettra d'avoir un gain maximal. La courbe de la figure 9 extraire de la fiche technique, montre le gain en fonction de la tension contrôlable. On remarque que, pour avoir un gain proche de +40dB, il faut une tension de contrôle proche de **2v (1.9v)**.

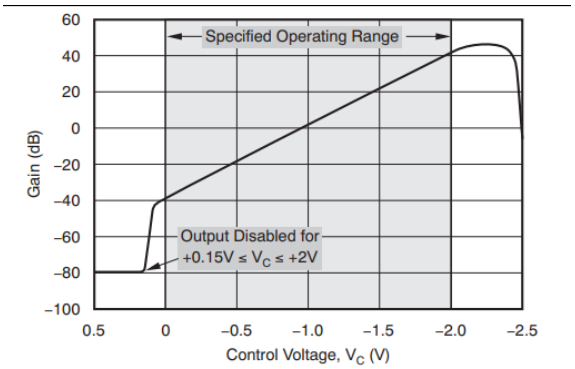


Figure 9: Gain vs tension contrôlable

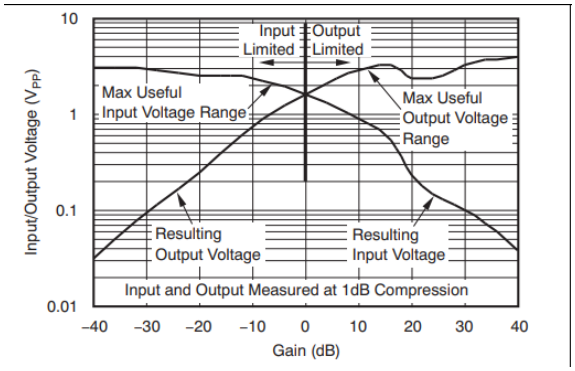


Figure 10: Plage de tension d'entrée et sortie vs Gain

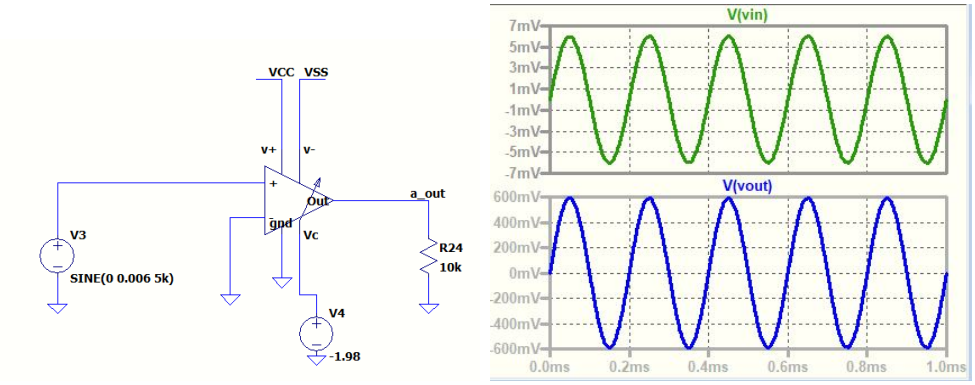


Figure 11: Schéma de montage et résultat de simulation pour vin=6mv

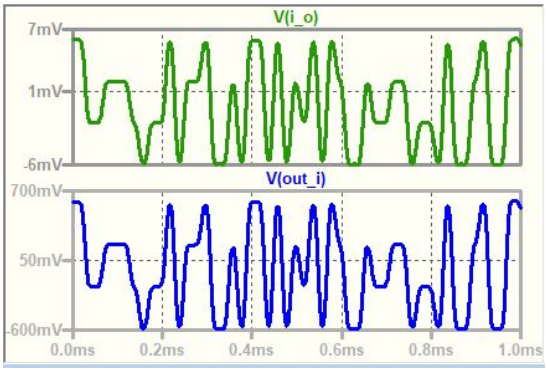


Figure 12: Amplification des 4 niveaux de tension.

Le montage réalisé sur LTSpice comme décrit dans la fiche technique et les simulations faites sur LTspice ont donné le résultat obtenu à la figure 11). On constate que la tension d'entrée (en verte qui est de **6mv**) est amplifiée (tension en bleu, **600mv**). On calcule le gain **G = 20log(Vs/Ve)= 20*600/6 = 40dB** qui est un gain maximal. Le Résultat de simulation pour les niveaux de tension : **6,2,-2 et 6mv** en entrée du filtre passe bas est illustré sur la figure 12.

Le modulateur MAX2021 a quatre entrées, du coup faut transformer la sortie de l'amplificateur à gain variable en deux tensions déphasées de 180 degré. C'est le but de l'utilisation de l'amplificateur différentielle THS4505(ou du AD8137).

2.4 Mise en forme des signaux de la bande de base pour le modulateur IQ

Le modulateur IQ utilise des entrées différentielles pour des voies I et Q. Du coup on utilise l'amplificateur différentiel à la sortie de l'amplificateur à gain variable pour générer deux tensions en opposition de phase à partir d'une seule tension en entrée. Ces tensions seront par la suite directement connectées à l'entrée du MAX2021.Nous avons utilisé l'AD8137 à la place du THS4505, car le THS4505 ne fonctionne pas avec notre version LTSpice. L'AD8137 est un amplificateur différentiel à faible consommation, avec une grande plage d'alimentation allant de 2.7v à 12v.Dans notre projet, nous travaillons avec une tension d'alimentation de +/-5v. Donc nous allons faire fonctionner notre amplificateur différentielle sous -/+5v. Le modulateur IQ MAX2021 peut fonctionner sous une plage de fréquence RF de 750MHz à 1200MHz. D'après la fiche technique, il doit être câblé comme le montre la figure 13. D'après la figure 13, **u3 < 4Vpp**, en appliquant le pont diviseur de tension, **u2 < 11.5V**. Si on prend **u2 = 4.84v** (on doit rester en dessous de 10v, car notre ampli fonctionne sous +/-5v), alors pour **u1=0.6v**(tension de sortie du VCA810), on aura une amplification de **G = 4.84/0.6=8.07**.

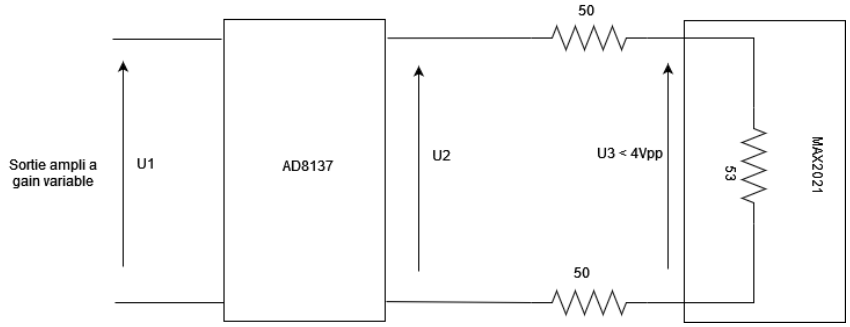


Figure 13: Schéma simplifié de l'amplificateur différentiel.

En considérant le montage de la figure 14 (extraire de la fiche technique), pour **R27=R26=8.2k et R24=R25=1k**, on a un gain **G=R27/R24=R26/R25=8.2**, ce gain est proche de ce qu'on recherche.

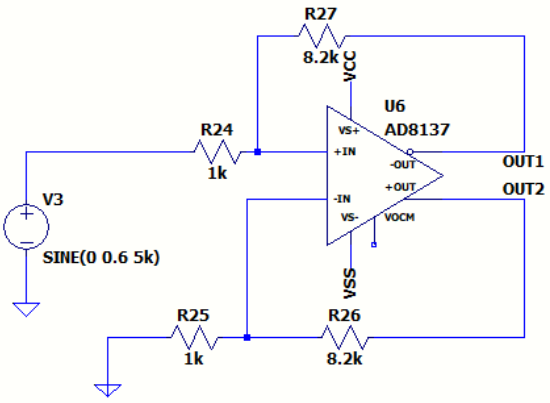


Figure 14: Schéma de montage sur LTSPICE de l'amplificateur différentiel.

Après simulation sur LTSpice, on observe une amplification G= 2.29*2/0.56=8.17. Pour être sûr de ne pas avoir de la distorsion, nous avons faire une simulation en prenant une tension sinusoïdale en entrée et en fin appliquer en entrée le signal à moduler.Les résultats obtenus sont illustrés sur la figure 15.

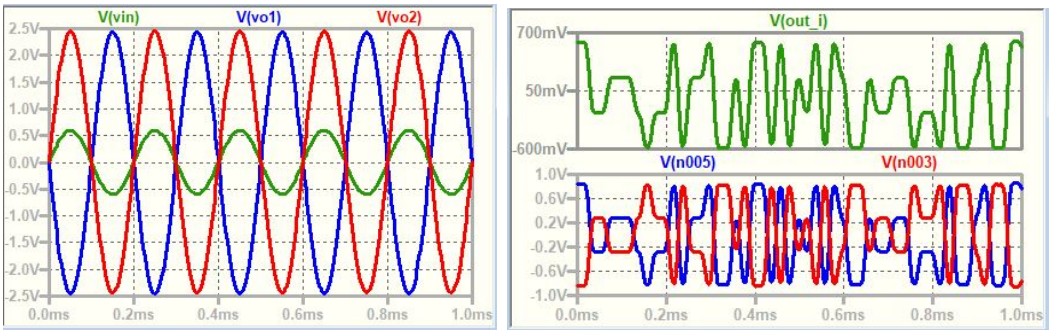


Figure 15: Résultat de simulation :gauche pour une tension sinusoidale et droite pour les 4 niveaux de tension.

Les deux résistances de 50 Ohm en série sur l'entrée du MAX2021 permettent l'adaptation. Le montage de la figure Fig. 16montre comment relier l'amplificateur différentiel au MAX2021.

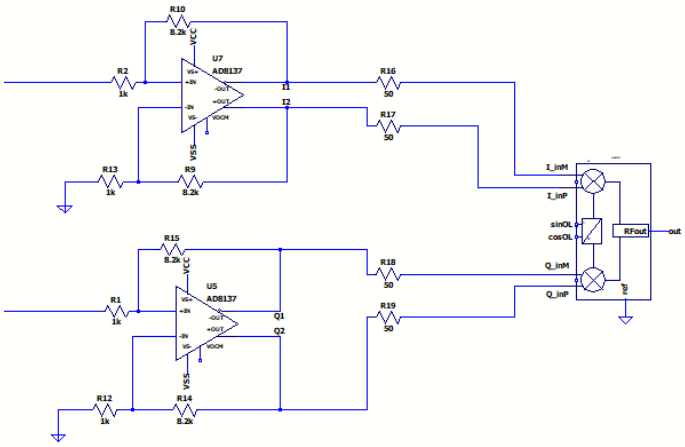


Figure 16: Montage de connection de l'ampli différentiel au MAX2021.

3 Simulations finales de l'émetteur

Le montage de la figure 17, est le circuit complet 16-QAM réalisé sur LTSpice.

On voit bien sur la figure 18 que le signal se trouve exactement sur 1MHz qui est la fréquence de la porteuse.

On constate que pour une fréquence de coupure du filtre passe bas petite, la largeur du spectre du signal modulé devient très petite. Par conséquent, on obtient une efficacité spectrale très loin de la valeur optimale qui est de l'ordre de 2 bit/sec/Hz pour un 16-QAM et 1 bit/sec/Hz pour un 4-QAM. Ce qui entrainera les pertes des informations (bits). Par ailleurs, lorsque la fréquence de coupure est très élevée, toute l'information sera transmise. En plus de l'information transmise, on aura d'autres informations parasites qui vont s'ajoutées sur l'information utile.

Le tableau 2, montre l'impact de la fréquence de coupure du filtre passe bas sur l'efficacité

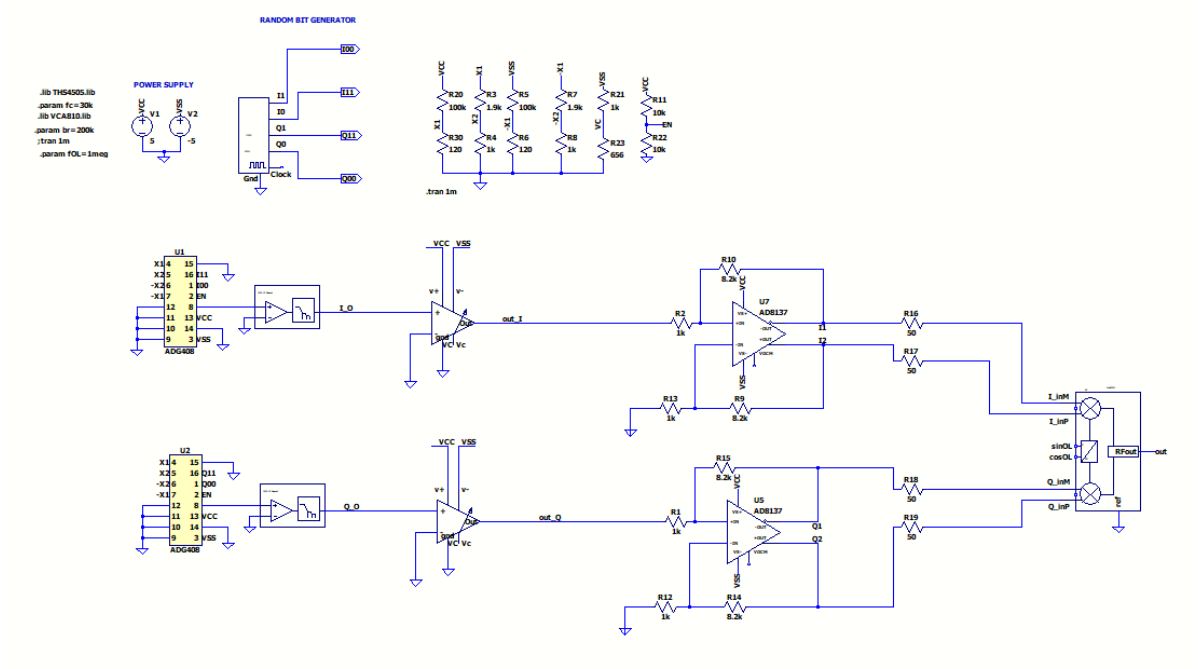


Figure 17: Montage complet de l'émetteur 16-QAM

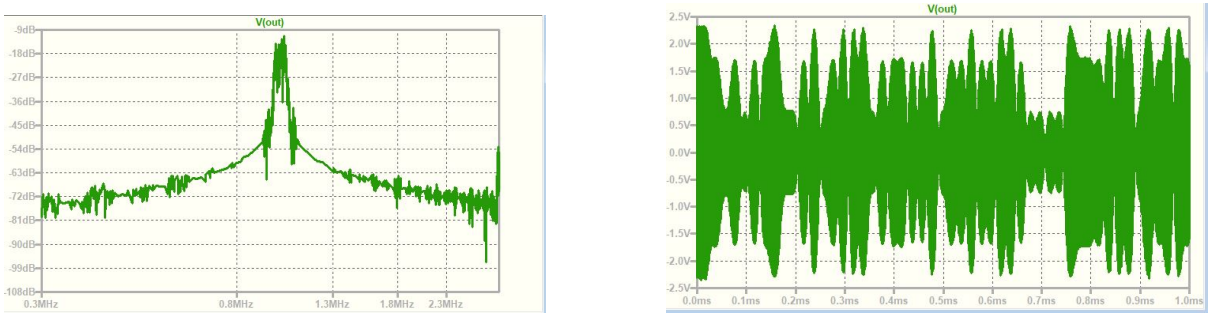


Figure 18: Sortie 16-QAM: droite sortie temporelle et gauche le spectre.

Table 2: Impact de la fréquence de coupure du filtre passe bas sur l'efficacité spectrale du signal

Fc KHz	**1/Tb - bit/sec**	**η bit/sec/Hz**
10	200k	9
25	200k	2.5
30	200k	2
35	200k	2

spectrale, plus la frequence de coupure dimunue, l'efficacité spectrale augmente, ce qui fait perdre des informations.
Les figures 19 illustrent l'impact de la variation de la frequence de coupure(30,10 et 20KHz) sur le pectre du signal module.On constate que l'image de gauche a toute l'information a transmettre alors que les deux figures de droite n'ont pas la totalité de l'information à transmettre. Le spectre du signal module et le montage permettant d'obtenir un NRZ-2 sont illustrés sur la figure 20.Pour obtenir un 4-QAM, il faut juste

modifier les entrées du multiplexeur analogique à deux entrée et une seule adresse.

Figure 19: Spectre du signal modulé pour fc= 30,20, et 10KHz.

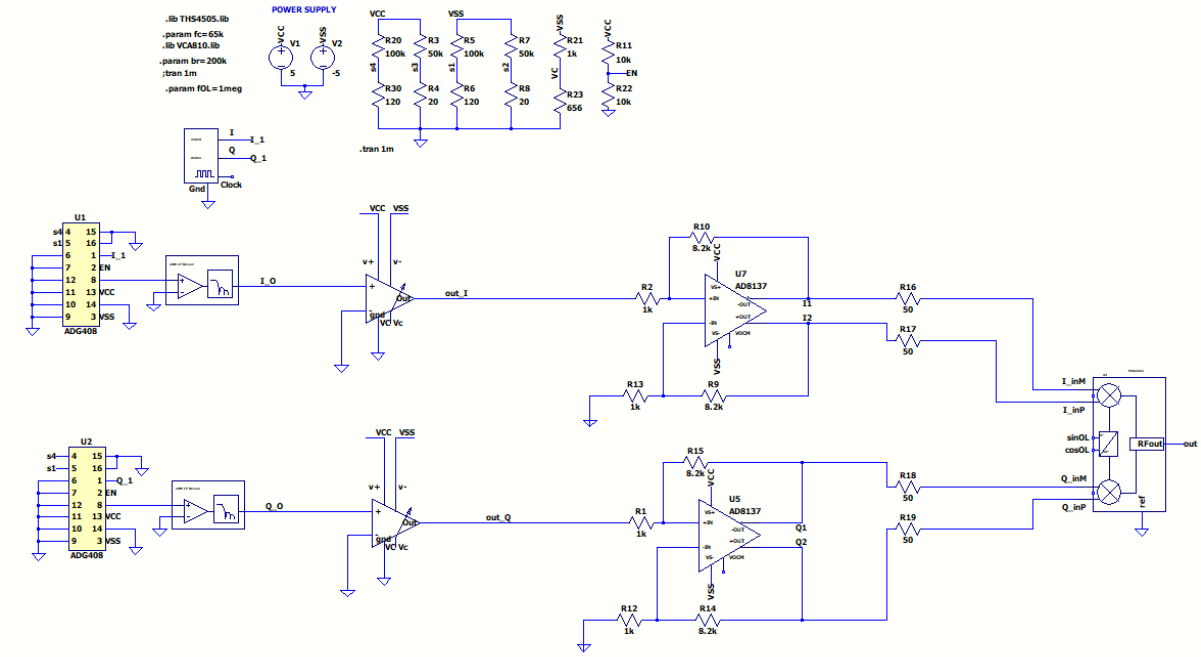


Figure 20: Montage complet de l'émetteur 4-QAM

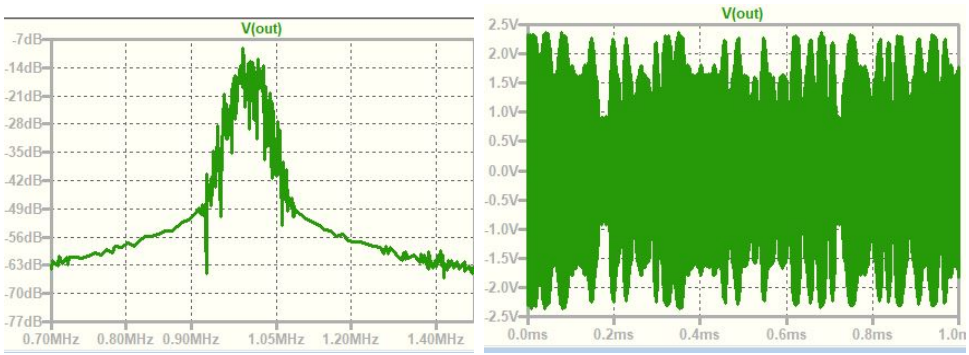


Figure 21: Sortie 4-QAM: droite sortie temporelle et gauche le spectre.

A partir de la figure 21, on trouve une efficacité spectrale de **200kb / (1.1-0.9) *1000=1 bit/sec/Hz**.

4 caractérisation de la PLL ADF4360-7

L'objectif de cette partie est la mise en œuvre de la PLL et la caractérisation complète de celle si, en programmant sur une carte ARDUINO UNO afin de générer la porteuse grâce au port SPI autour de 900 Mhz. La PLL est un circuit intégré dont les registres internes peuvent être configurer à travers une communication (SPI) avec le microcontrôleur. Ce circuit intégré comporte trois registres : le registre R, contrôle(C) et enfin le registre N. Cette PLL fonction en mode 3 (SPI). Pour que le microcontrôleur puisse communiquer correctement avec cette PLL, l'on doit respecter la séquence d'écriture des registres dans l'ordre suivant : R, C et N. Ces registres sont des registres à 24 bits, du coup pour les configurer, il faut envoyer 8 fois 3 bits, car nous avons choisir d'envoyer 8 bits à chaque transfert de donnée.

Les tableaux decrivants les bits et les valeurs à écrire dans les registres afin de verrouiller la PLL sont illustrés sur la figure 22 23 24.

R COUNTER LATCH

RESERVED	RESERVED	BAND SELECT CLOCK		TEST MODE BIT	LOCK DETECT PRECISION	ANTI-BACKLASH PULSE WIDTH		14-BIT REFERENCE COUNTER												CONTROL BITS	
DB23	DB22	DB21	DB20	DB19	DB18	DB17	DB16	DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
RSV	RSV	BSC2	BSC1	TMB	LDP	ABP2	ABP1	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	C2 (0)	C1 (1)

Figure 22: Registre R : **0, 0, 0xC9**

CONTROL LATCH

PRESCALER VALUE		POWER-DOWN 2	POWER-DOWN 1	CURRENT SETTING 2			CURRENT SETTING 1			OUTPUT POWER LEVEL		MUTE-TILL-LD	CP GAIN	CP THREE-STATE	PHASE DETECTOR POLARITY	MUXOUT CONTROL			COUNTER RESET	CORE POWER LEVEL		CONTROL BITS	
DB23	DB22	DB21	DB20	DB19	DB18	DB17	DB16	DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
P2	P1	PD2	PD1	CPI6	CPI5	CPI4	CPI3	CPI2	CPI1	PL2	PL1	MTLD	CPG	CP	PDP	M3	M2	M1	CR	PC2	PC1	C2 (0)	C1 (0)

Figure 23: Registre de controle:**0x4F,0xF9,0x20**

N COUNTER LATCH

DIVIDE-BY-2 SELECT	DIVIDE-BY2	CP GAIN	13-BIT B COUNTER													RESERVED	5-BIT A COUNTER					CONTROL BITS	
DB23	DB22	DB21	DB20	DB19	DB18	DB17	DB16	DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DIVSEL	DIV2	CPG	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	RSV	A5	A4	A3	A2	A1	C2 (1)	C1 (0)

Figure 24: Registre R : **0x1,0x19 ,0x12 (pour A=4 et B = 281)**

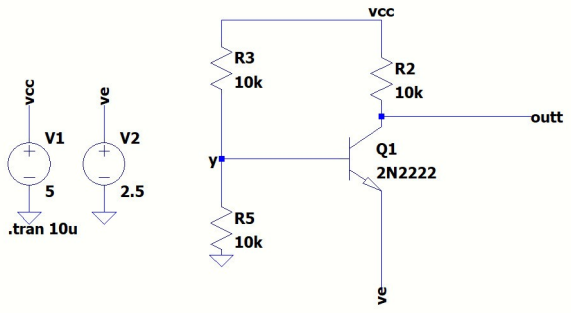


Figure 25: translateur de niveau à base de 2N2222

La configuration des pins du microcontrôleur et la carte de la PLL est la suivante:
-le vert représente la donnée qui est relier à la broche D11(MOSI) de l'arduino

-le bleu représente l'horloge qui est relier à la broche D13(CLK) de l'arduino
-le jaune représente LE qui est relier à la broche SS0 (D10) de l'arduino
-et enfin le noir représente la masse qui est relié à la masse du arduino.

Nous avons utilisé l'entrée digitale (digitalRead) pour déterminer l'état du MUXOUT de la PLL, du coup nous avons implémenté le translateur de niveau de tension de la figure 25.En effet, ce circuit permet de convertir la sortie MUXOUT (3.3v) a 5v.

4.1 Travail effectué

4.1.1 Plage de fonctionnement en fréquence

En jouant sur les valeurs du compteur A, et B, on parvient à trouver la fréquence minimale et maximal de verrouillage de la PLL. Soit la formule $Fosc = (BP + A)Fref/R$ (1) . On calcule R = 50 pour Fosc = 900MHz, Fref = 10MHz, le pas de 200kHz. Si on prend A = 4, alors B = 281 et p=16. L'étape suivante consiste à trouver les valeurs à mettre dans les différents registres. Ces valeurs sont énumérées en figure 22 23 24. Pour faciliter l'écriture dans les registres, nous avons développé un sous-programme qui permet d'envoyer directement 24 bits sur le MOSI illustrée sur la figure 26.Ce sous-programme prend en paramètre un tableau à trois éléments. De cette façon, on pourra incrémenter aisément B. Par la suite nous avons développé un masquage qui nous permet

```
void write24bits(byte tab[], byte len){

      for (byte i = 0; i<len ; i++){
          SPI.transfer(tab[i]);
       }
 }
```

Figure 26: Sous-programme pour transferer 24bits.

d'incrémenter directement A et B figure 27

```
byte  X_5 = ((B&0x1F00)>> 8);     //EXTRACT THE 5 FIRST BITS
byte  X_8 =   B&0xFF;             //EXTRACT THE LAST 8 BITS
byte   result_A  = (A<<2)|0x02;

tab3[0] = X_5;
tab3[1] = X_8;
tab3[2] = result_A;
```

Figure 27: Programme de masquage des 5 et 8 bits de B

.

Ce masquage permet de récupérer les 5 bits de poids fort de B, ainsi que les 8 bits de poids faible. A présent il ne reste qu'à incrémenter B et fixer A = 4. Apres compilation nous avons relevé les différentes valeurs de B qui sont de **231** à **316** avec $Bmin = 231$ et $Bmax = 316$. En appliquant la formule (1), on trouve **Fmin = 740MHz et Fmax= 1012MHz**.

4.1.2 Temps de verrouillage

Pour trouver le temps de verrouillage de la PLL on a développé un code avec un timer et une interruption. Lorsqu'on finit l'initialisation des registres dans la fonction setup, on déclenche le timer. Une fois que le MUX OUT passe à 1, on désactive le timer et on sauvegarde la valeur de d'une variable count (voir code en annexe). Nous avons fixé la période du timer à 16us. A chaque 16us, on entre dans le sous-programme de

l'interruption et incrémente une variable count (voir code en ANNEXE) et la fin (lorsque le MUX OUT passe à 1) on multiplie count par 16us pour trouver le temps de verrouillage

Par la suite nous avons estimé le temps de verrouillage en utilisant un oscilloscope comme le montre la figure 28. Le temps de verrouillage à l'oscilloscope est de **40us**. Avec notre programme, nous avons trouvé un count = 3, or notre période est fixée a 16us, du coup on trouve un t=16*3=**48us**.

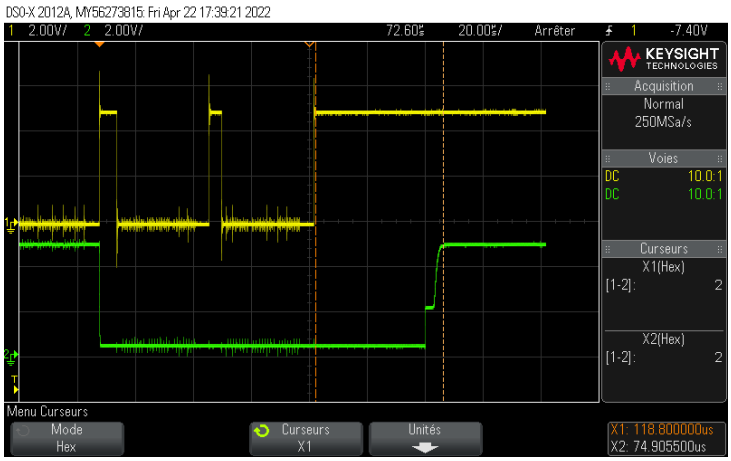


Figure 28: temps de verrouillage de la PLL :le jaune représente la pin LE et le vert est la sortie du mux out.

4.1.3 Précision de la PLL

En Utilisant l'analyseur de spectre on trouve la précision en fréquence de notre PLL. Nous avons pris deux valeur de B qui se suivent, en faisant la différence des deux fréquences obtenues a partir des deux B, on trouve une valeur de **190KHz**, qui est à peu près égale à **200KHz**.La figure 29 illustre la spectre pour B=240 et A=4.

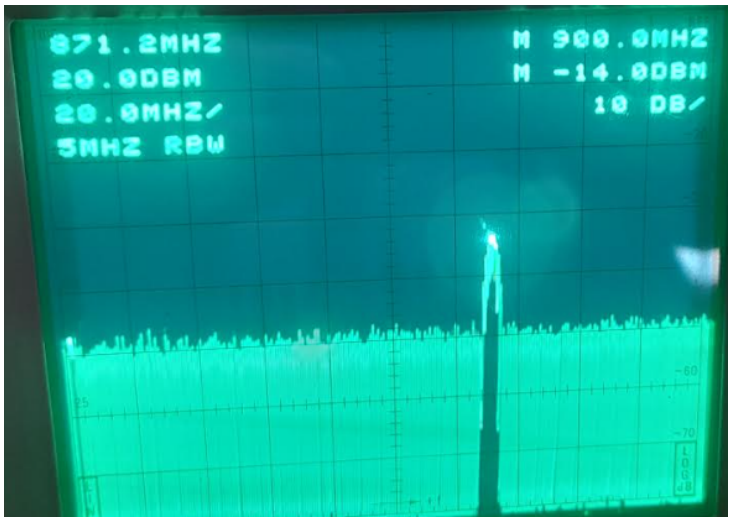


Figure 29: Spectre pour B=240, A=4

4.1.4 Extraction des puissances émises

Le choix de la puissance émise se fait par la combinaison des bits DB13, DB12 du registre de contrôle. Pour **0xD9,0xC9,0xE9 et 0xF9**, on trouve respectivement **-14.3, -15.0,-**

13.7 et -14.3dBm. Ces résultats sont différents de ceux de la fiche technique à cause du vieillissement des composants.

5 Conclusion

En sommes, nous avons réalisé en premier partie de ce projet un émetteur RF. L'implémentation de ce système sur LTSPICE, nous a éclaircie le fonctionnement d'un 16-QAM et 4-QAM. Nous avons appris comment dimensionner un circuit électronique tout en respectant les contraintes imposées par la fiche technique des composants électroniques. Par ailleurs on n'a vu l'impact du filtre passe bas sur le signal modulé. En deuxième partie de ce projet, nous avons caractérisé la PLL DF4360-7. On n'a bien compris comment fonctionne une communication SPI, nous avons déterminé la fréquence minimale et maximal de verrouillage de la PLL, le temps de verrouillage, les puissances etc. Cette PLL peut être utilisé pour la démodulation du signal transmise par l'émetteur RF, en tant que suiveur de fréquence, et par conséquence on pourra extraire tous les bits envoyés par l'émetteur.

6 Annexe

```
#include <SPI.h>
#define SS0 10  // affectation de la broche SS0
#define MUX_PIN 3   // affectation de la broche MUX_PIN

  byte tab1[3]={0x0,0x0,0xC9};      // R : COUNTER LATCH
  byte tab2[3]={0x4F,0xF9,0x20};    // Control
  byte tab3[3]={0x1,0x19,0x12};     // N : A = 4  B= 281
  byte flag = 1;
  //ordre R, C 10, N --- 1,3,2
  int A =4,B = 281;
  int MUX = 0;
  int count = 0;
  int dValue = 0;
void setup() {

   Serial.begin(9600);
   pinMode(SS0,OUTPUT);
   //pinMode(A0,INPUT);
   pinMode(MUX_PIN,INPUT);
   digitalWrite(SS0, HIGH);
   SPI.beginTransaction(SPISettings(1000000,MSBFIRST,SPI_MODE3));  // try with 100 000

 noInterrupts();
 // start the SPI library:
 SPI.begin(); //initialisation de  la communication sur le bus

TCCR1A = 0b00000000;
TCCR1B = 0b00001100;  //256
TIMSK1 = 0b00000010;

 //R
 digitalWrite(SS0, LOW);
 write24bits(tab1, 3);
 digitalWrite(SS0, HIGH);

 //C;
 digitalWrite(SS0, LOW);
 write24bits(tab2, 3);
 digitalWrite(SS0, HIGH);


 delay(10);
 //N
 digitalWrite(SS0, LOW);
 write24bits(tab3, 3);
```

```
    //N
    digitalWrite(SS0, LOW);
    write24bits(tab3, 3);
    digitalWrite(SS0, HIGH);
    SPI.endTransaction(); //arrête la communication.
    TCNT1  = 0;
    OCR1A  = 1;
    interrupts();
  }

  //
  ISR(TIMER1_COMPA_vect)
  {
  count++;

  }
  if(digitalRead(MUX_PIN)==1 && flag ){
  //
    noInterrupts();
    Serial.println(TCNT1);
    Serial.println(count);
     count = 0;
     flag = 0;
     delay(10);

    }
  //
        byte  X_5 = ((B&0x1F00)>> 8);     //EXTRACT THE 5 FIRST BITS
        byte  X_8 =   B&0xFF;             //EXTRACT THE LAST 8 BITS
        byte   result_A  = (A<<2)|0x02;
        tab3[0] = X_5;
        tab3[1] = X_8;
         delay(10);
         digitalWrite(SS0, LOW);
         write24bits(tab3, 3);
         digitalWrite(SS0, HIGH);
  while(1);

}

 void write24bits(byte tab[], byte len){

      for (byte i = 0; i<len ; i++){
          SPI.transfer(tab[i]);  //envoie l'octet sur la ligne MOSI ou reçoit l'octet sur la ligne MISO
       }
  }
```