



Rapport d'activités fin avril mi-mai 2022

TUELEAU Tom

28 mai 2022



Table des matières

1	Introduction	3
2	Finalisation du montage	3
2.1	Montage	3
2.2	Installation dans la ruche	3
2.3	Résultat	4
2.4	Amélioration	5
3	Programmation	6
3.1	Programmation des capteurs	6
3.1.1	Capteur de température et humidité	6
3.1.2	Capteur de vibration	6
3.1.3	Traitement des données	7
3.2	Programmation réseaux	8
3.2.1	Topologie du réseaux	8
3.2.2	Liaisons MQTT	8
3.2.3	Liaisons UDP	9
4	Conclusion	9

1 Introduction

Ce document a pour objectif de faire un état d'avancement du stage. Celui-ci résumera donc le travail fait lors de la fin du mois d'avril et la première moitié du mois de mai.

Dans un premier temps, je reviendrais sur le montage amplificateur vu lors du dernier rapport et vous montrerais les résultats obtenus.

Une seconde partie présentera les programmes créés afin de récolter et traiter les données des capteurs. Une troisième partie traitera de l'envoi des données entre les différents éléments du système.

Enfin, je conclurai sur le travail effectué et les difficultés rencontrées.

2 Finalisation du montage

Lors des précédents rapports¹, je vous ai exposé mes recherches et résultats quant au dimensionnement d'un montage amplificateur. Ce besoin s'était fait ressentir quand je m'étais rendu compte que le signal émis par le piézo-électrique n'arrivait pas à être capté par l'Arduino. Dans cette partie nous verrons tout d'abord la finalisation du montage et dans un second temps nous verrons l'installation de celui-ci dans le rucher.

2.1 Montage

Lors de cette semaine j'ai pu effectuer le prototype final incluant l'amplification du signal, le piézo-électrique, le capteur de température et d'humidité (Si7021) et l'Arduino. Vous pouvez voir un schéma complet Figure 1.

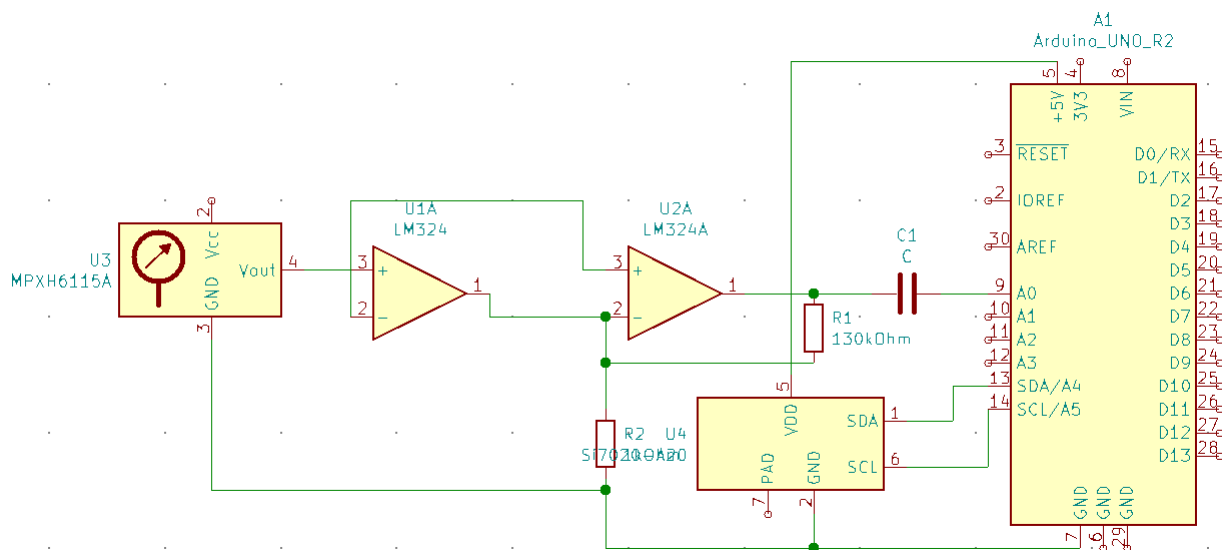


FIGURE 1 – Schéma montage final

Comme nous le verrons lors de la partie programmation, j'arrive à récupérer le signal envoyé par le piézo-électrique avec l'Arduino. Cela va donc me permettre d'échantillonner celui-ci et de le traiter.

2.2 Installation dans la ruche

L'objectif étant de récolter les données dans la ruche, j'ai dû installer les capteurs dans celle-ci. Comme vous pouvez le voir Figure 2, j'ai glissé les différents capteurs dans la ruche et leur ai soudé des fils assez longs pour les brancher par la suite sur un microcontrôleur.

1. Voir "Rapport d'activités du 11 et 18 avril 2022", partie 4, page 7



FIGURE 2 – Capteurs dans la ruche

2.3 Résultat

Après avoir échantillonner le signal obtenue par le piezo et l'avoir envoyer aux serveur de traitement, j'ai pu obtenir le graphique Figure 3².

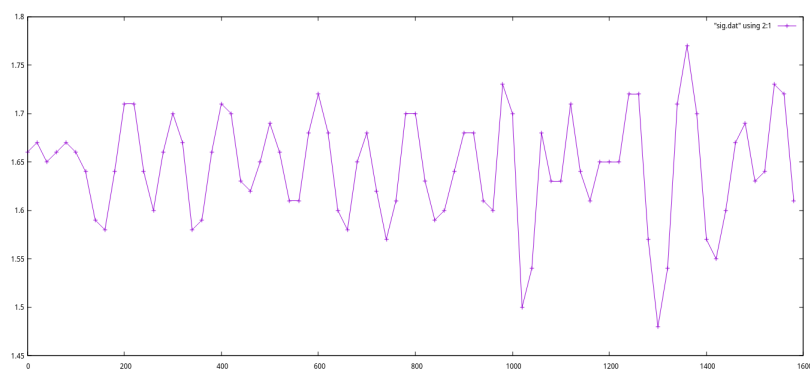


FIGURE 3 – Graph signale piezo

2. Abscisse : temps en ms Ordonné : tension en V

2.4 Amélioration

Maintenant que ce montage est fonctionnel, j'aimerais pouvoir lui apporter des améliorations. La première serait la création d'un PCB pour le montage amplificateur afin de rendre le montage plus professionnel. La seconde serait la création d'un boîtier pouvant le contenir lui et l'Arduino.

3 Programmation

Tout au long de la dernière semaine, j'ai eu à programmer plusieurs fonctionnalités. Dans cette partie je vais donc commencer par vous présenter le programme effectué pour récolter les données des capteurs et effectuer leur traitement. Une deuxième partie abordera la programmation de la mise en communication des différents composants du système.

3.1 Programmation des capteurs

Pour ma première mission j'avais trois données à récupérer. Celle-ci sont l'humidité, la température et les vibrations. Afin de récolter ces données j'ai choisi en tout trois capteurs. Un capteur piézo-électrique, un capteur de température et d'humidité (Si7021) et un microphone (INMP441). Dans un premier temps je vous parlerai du capteur de température et d'humidité. Une seconde partie abordera la programmation du capteur de vibration. Enfin une dernière partie abordera le traitement des données.

3.1.1 Capteur de température et humidité

Comme indiqué précédemment le capteur utilisé pour ces données est le Si7021. Étant donné que le microcontrôleur que j'utilise est un Arduino, il existe des bibliothèques donnant accès à des fonctions permettant d'obtenir les données voulues.

```

1  #include <Adafruit_Si7021.h>
2
3  Adafruit_Si7021 sensor = Adafruit_Si7021();
4
5  void callback(char* topic, byte* payload, unsigned int length) {}
6
7  void setup() {
8      Serial.begin(9600);
9      if (!sensor.begin()) {
10         Serial.println("Did not find Si7021 sensor!");
11         while (true)
12             ;
13     }
14 }
15
16 void loop() {
17     double hum,temp;
18     char Ctemp[4], Chum[4];
19     hum = sensor.readHumidity();
20     temp = sensor.readTemperature();
21     dtostrf(temp,2,2,Ctemp);
22     dtostrf(hum,2,2,Chum);
23     delay(1000);
24 }
25

```

Les fonctions principales de ce programme sont, "sensor.begin" qui initialise la liaison entre l'Arduino et le capteur, readHumidity et readTemperature permettent comme leurs noms l'indiquent de récupérer la température et l'humidité mesurées par le capteur. Les valeurs retournées par ces deux fonctions sont de type "double". Afin de pouvoir les envoyer, je les convertis en chaîne de caractères à l'aide de la fonction "dtostrf".

3.1.2 Capteur de vibration

Après avoir amplifié le signal du piézo-électrique, j'ai pu me consacrer à l'acquisition des données de celui-ci. Afin de faciliter son acquisition j'ai créé une fonction "udpSendVibration" prenant la quan-

titer d'échantillon souhaités, le nombre de decimal et la période d'échantillonnage (en milli seconde).

```

1
2 void udpSendVibration(int nbEchantillon, int nbVirgule, int periodeEchantillageMs)
3 {
4
5     int nbChar = nbVirgule+2;
6     char tmp[nbEchantillon][nbChar];
7     char data2[nbEchantillon*nbChar];
8     double centsValeurs[nbEchantillon];
9     int i,y;
10    for(i=0;i<nbEchantillon;i++){
11        centsValeurs[i]=analogRead(piezo);
12        centsValeurs[i]=(centsValeurs[i]*5)/1023;
13        delay(periodeEchantillageMs);
14    }
15    delay(2000);
16    for(i=0;i<nbEchantillon;i++){
17        dtostrf(centsValeurs[i],nbChar,nbVirgule,tmp[i]);
18        for(y=0;y<nbChar;y++){
19            data2[(nbChar*i)+y]=tmp[i][y];
20            Serial.println(data2[(i*nbChar)+y]);
21        }
22    }
23
24 }
```

3.1.3 Traitement des données

Une fois le signal obtenue il fallait l'analyser en fréquence. Pour cela j'étais dans un premier temps partie pour traiter directement sur l'Arduino en effectuant une transformée de Fourier du signal. Cependant les capacités de calcul de l'Arduino étant limitées et ayant observé une certaine lenteur à effectuer cette opération j'ai décidé d'effectuer le système suivant. J'envoie les échantillons à un ordinateur distant qui effectuera le traitement.

J'ai donc par la suite écrit un programme utilisant la bibliothèque "fftw". Celle-ci proposant de multiples fonctions basées sur l'algorithme FFT (Fast Fourier Transform), permettant d'effectuer une transformée de Fourier sur les signaux numérisés.

```

1
2 unsigned int N = 20;
3 int i;
4 double *out = malloc(sizeof(double) * N);
5 double list[]={3.920000,0.160000,5.630000,4.750000,0.370000,2.010000,
6 5.870000,6.730000,9.870000,4.340000,9.950000,5.460000,8.420000,3.120000,
7 3.670000,4.870000,3.280000,6.050000,4.650000,0.500000};
8 double * values;
9 values = malloc(sizeof(double)*N);
10 fftw_plan p;
11 p = fftw_plan_r2r_1d(N,values,out,FFTW_HC2R,FFTW_MEASURE);
12
13 for(i=0;i<20;i++){
14     values[i]=list[i];
15 }
16
17 fftw_execute(p);
18 for(i=0;i<20;i++){
```

```

19     printf("%f ",out[i]);
20 }
21 printf("\n");
22 fftw_destroy_plan(p);
23 free(out);

```

Dans l'exemple ci-dessus on effectue la transformation de Fourier du signal composé des points de la liste "list".

3.2 Programmation réseaux

Afin d'acheminer les informations jusqu'aux différentes composantes (site web, serveur) j'ai eu à programmer plusieurs liaisons réseaux. Je vais donc commencer par vous présenter la topologie de celui-ci avec les différents éléments le composant ainsi que leur rôle. Dans un second temps je détaillerai la liaison MQTT reliant la base de données et l'Arduino. Une dernière partie traitera de la communication des données au serveur.

3.2.1 Topologie du réseau

Comme dit précédemment le système est composé de plusieurs éléments. Tout d'abord l'Arduino recueille les données et les envoie aux autres composantes. La température et l'humidité sont envoyées via MQTT vers le serveur tandis que les données du piezo sont envoyées via UDP. Une fois traitées ces dernières sont aussi envoyées sur un topic MQTT.

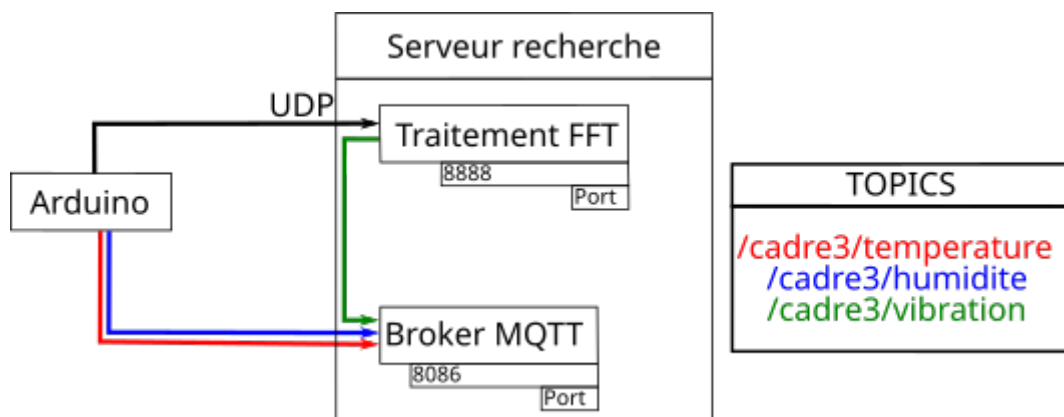


FIGURE 4 – Schema Réseaux

3.2.2 Liaisons MQTT

Afin d'envoyer les données du microcontrôleur vers le serveur j'ai choisi le protocole MQTT et ce pour deux raisons. La première est que le projet sera amené à évoluer et contiendra plusieurs capteurs récoltant la même donnée à des endroits différents de la ruche. Ainsi avec le système de topics, on pourra identifier facilement ces différents capteurs. La deuxième est que la solution envisagée pour le site web, et la base de données, est très facilement couplable avec ce protocole. Ci-dessous on peut voir le code utilisé pour envoyer sur les topics "cadre3/temperature" et "cadre3/humidite".

```

1  ...
2  if(!client.publish("cadre3/temperature",Ctemp))
3  {
4      Serial.println("error");
5  }
6  if(!client.publish("cadre3/humidite",Chum))
7  {
8      Serial.println("error");
9  }

```


10 ...

3.2.3 Liaisons UDP

Afin de transmettre les données du piezo au serveur, j'ai choisi d'effectuer une liaison UDP. J'ai choisi ce protocole car la liaison n'a pas besoin de haute fidélité. L'Arduino envoie donc sur le serveur les données du piezo. Le programme ci-dessous crée une socket et attend que l'Arduino envoie ces données.

```

1
2  ...
3
4  int sockfd, len, n, echantillon, nbchar;
5  char buffer[MAXLINE];
6  struct sockaddr_in servaddr, cliaddr;
7
8  if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
9      perror("Erreur lors de la creation de la socket");
10     exit(EXIT_FAILURE);
11 }
12
13 memset(&servaddr, 0, sizeof(servaddr));
14 memset(&cliaddr, 0, sizeof(cliaddr));
15
16 servaddr.sin_family = AF_INET;
17 servaddr.sin_addr.s_addr = INADDR_ANY;
18 servaddr.sin_port = htons(PORT);
19
20 if ( bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0 )
21 {
22     perror("bind failed");
23     exit(EXIT_FAILURE);
24 }
25
26 len = sizeof(cliaddr); //len is value/result
27 for(;;){
28     n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, ( struct sockaddr *) &cliaddr, &
29     len);
29     buffer[n] = '\0';
30     sendto(sockfd, "ACK", strlen("ACK"), MSG_CONFIRM, (const struct sockaddr *)&cliaddr, sizeof(
31     cliaddr));
32 }
33
34 ...

```

4 Conclusion

En conclusion, maintenant que les données ont été récoltées et acheminées, il faut pouvoir les afficher sur un site web. Il me reste aussi à effectuer un montage final plus propre pour la partie électronique du système.

Table des figures

1	Schéma montage final	3
2	Capteurs dans la ruche	4
3	Graph signale piezo	4
4	Schema Réseaux	8