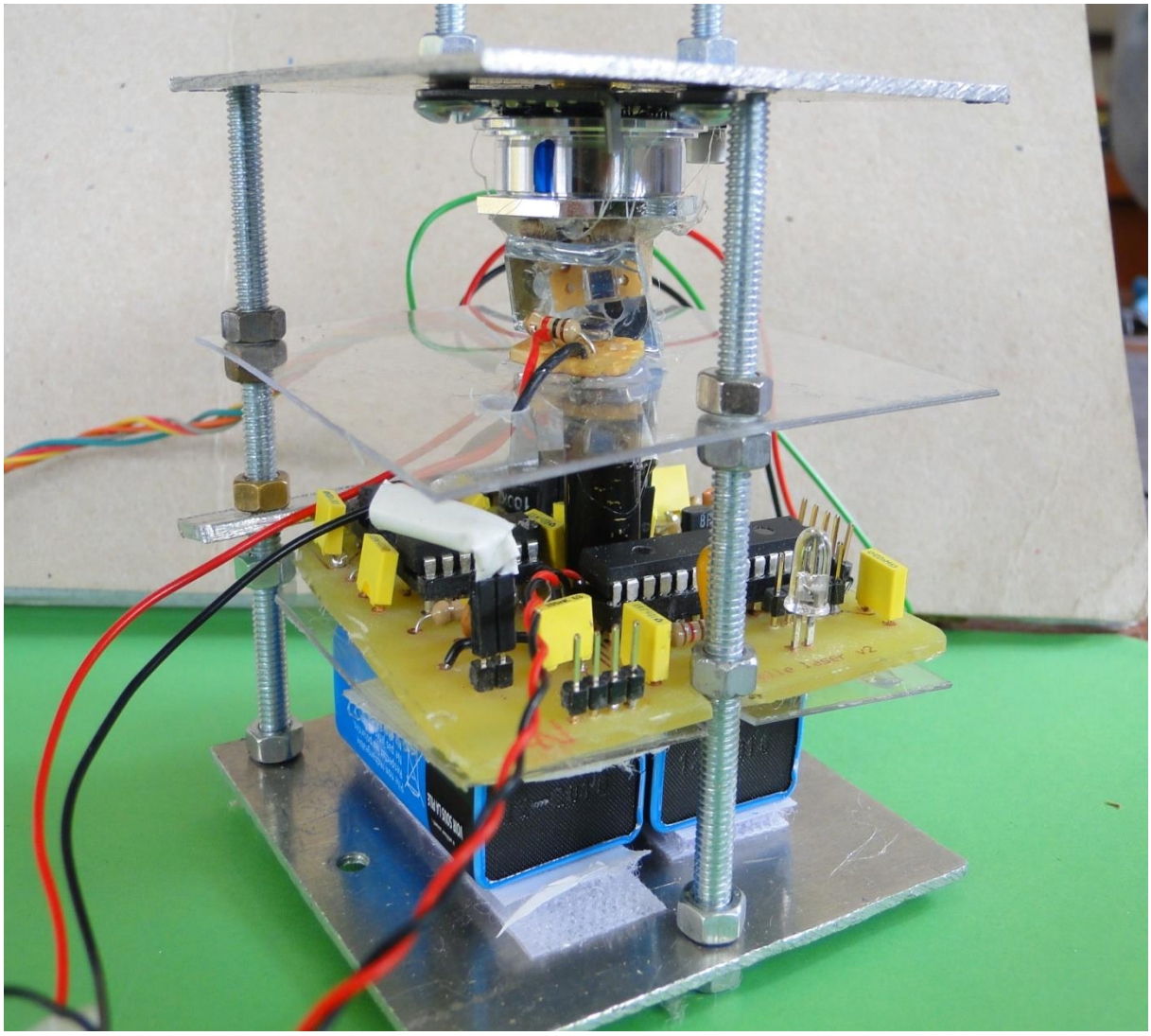
Aziz BELHASSAN
Eric FELTRIN
Sahbi THAIRI
Adrien THIBEAUD

Projet de Groupe n°6 :

Coupe de Robotique

Rapport Final

22 Mai 2013



SOMMAIRE

I- Introduction :

Nous réalisons le projet de groupe n°6, intitulé « Coupe de Robotique ». Le but de ce projet est de réaliser, en partenariat avec le club de robotique de Phelma, un système permettant de localiser un robot adverse sur la table de jeu de la coupe de France de robotique. En effet, d'après le règlement de la coupe de robotique les 2 robots présents sur la table de jeu ne doivent pas entrer en collision. En supposant que l'on sache à tout moment la position de notre robot, il nous faut donc idéalement connaitre la position du robot adversaire sur la table afin de pouvoir l'éviter.

Le système de détection doit être composé d'une tourelle sur le robot adverse et d'un maximum de trois balises sur les bords du terrain.

Notre projet est donc soumis aux contraintes de la coupe de robotique, que ce soit en termes de dimensionnement du système, de sécurité…

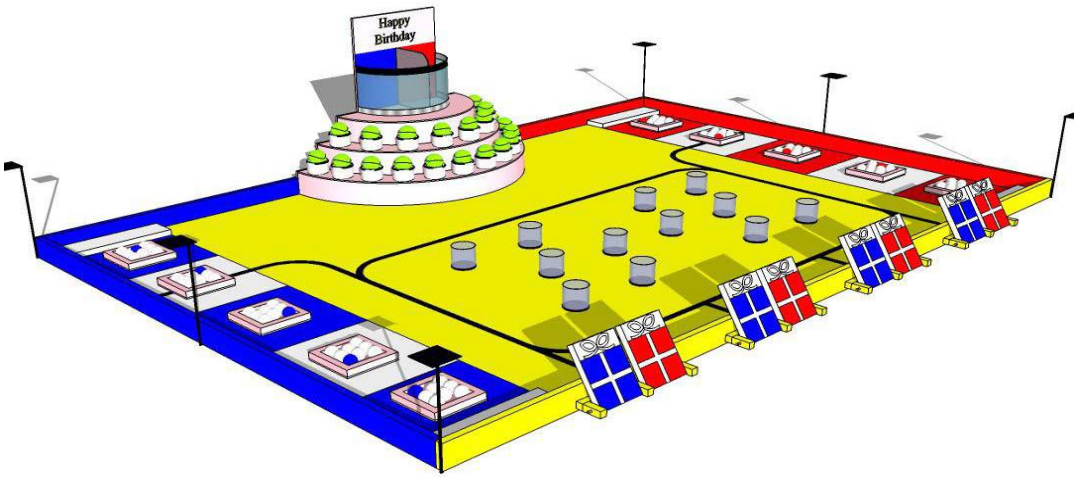


Figure 1 : Image 3D de la table de jeu

II. Cahier des charges :

L'étude du cahier des charges est une partie indispensable avant de passer à la partie purement technique du projet. Nous permettant ainsi de définir le cadre de réalisation de notre projet.

II.A) Objectif général :

L'objectif principal du projet est d'éviter une collision avec le robot adverse. Cet objectif, nous l'avons traduit par « mesurer la position du robot » adverse sur la table de jeu avec précision.

II. B) Règles générales :

Concernant les balises de détection (actives et passives), le règlement impose les quelques règles suivantes :

1. Les balises ne doivent d'aucune manière ni gêner ni brouiller l'adversaire. Leurs mises en place ne doit entraver le bon déroulement de la partie. Elles doivent rester en place sur leur support durant toute la durée du match.

2. Trois supports de balises fixes sont disposés le long de chaque largeur de l'aire de jeu (Voir figure 1 plus haut). Ils sont de couleurs noires et placés à une hauteur de 350 mm du niveau de la piste (Voir Annexe figure1).

3. L'utilisation des balises est facultative et leur construction reste à la charge des équipes.

II.C) Balises fixes et Tourelle :

Cependant en plus de ces quelques règles générales, il existe des règles spécifiques concernant les balises fixes et la tourelle (=balise embarquée) :

1. **Dimensions maximales :** 80x80x80mm

2. **Autonomie de la balise :** alimentation sur batterie, pile…

3. **Communication :** Coder les signaux de communication pour éviter les interférences

4. **Laser :** Seuls sont tolérés les lasers de type classe 1 ou 1M

III. Solutions envisagées et solution retenue :

Nous venons de définir le cadre de travail de notre projet par la réalisation du cahier des charges. Notre but : « Eviter la collision des 2 robots sur la table de jeu » peut être réalises par plusieurs systèmes différents en combinant balises fixes et balise embarquée. La seconde phase du projet a donc consisté en la réalisation d'une étude des avantages et inconvénients des systèmes possibles.

III. A) Solutions envisagées :

Le tableau ci-dessous recense les 3 types de système de détection envisagés avec leur principe de mesures et leurs points forts/faibles.

Type de Système de détection	Ultrasons	Infrarouge	Laser
Mesures effectuées	Distances	Distances	Angles
Principe de mesures	Connaissant la vitesse du son dans l'air, on peut déterminer la distance séparant le robot des balises extérieures et donc la position du robot.	On émet de la lumière infrarouge dans toutes les directions.	On fait tourner à vitesse constante sur le robot adverse un laser. Le rayon laser est alors réfléchi à l'aide de catadioptres sur les balises extérieures. On peut alors déterminer la position du robot par triangulation.
Points Forts	1. Simplicité de la mesure 2.Coût assez faible	1. Simplicité du système 2.Coût assez faible	1. Grande précision 2. Peu de perturbation
Points Faibles	1. Beaucoup de perturbations pouvant entrainer une incapacité de mesure 2. Phénomènes d'échos	1. Faible précision 2.Sensible au fort éclairage	1. Complexité du calcul de triangulation 2. Le rayon laser doit rester parfaitement horizontale 3. Coût élevé du laser
Autre remarques	Le projet de groupe de l'année dernière c'est retrouvé dans l'incapacité d'acquérir une mesure. (beaucoup trop de perturbations)	Le système d'anti-collision de cette année repose sur l'utilisation de capteurs infrarouges. On a ainsi pu observer lors du 1e match que le fort éclairage saturé ces capteurs.	

III. B) Solution retenue :

Malgré une plus grande complexité, la solution retenue fût celle du **radar laser** pour sa grande précision et afin de pouvoir diminuer les problèmes dû aux perturbations extérieures. De plus toujours dans l'optique de minimiser les perturbations éventuelles et surtout le cas où notre adversaire utiliserais également ce type de radar (fréquent durant la coupe), nous avons opté plus précisément pour un **radar laser à détection synchrone**.

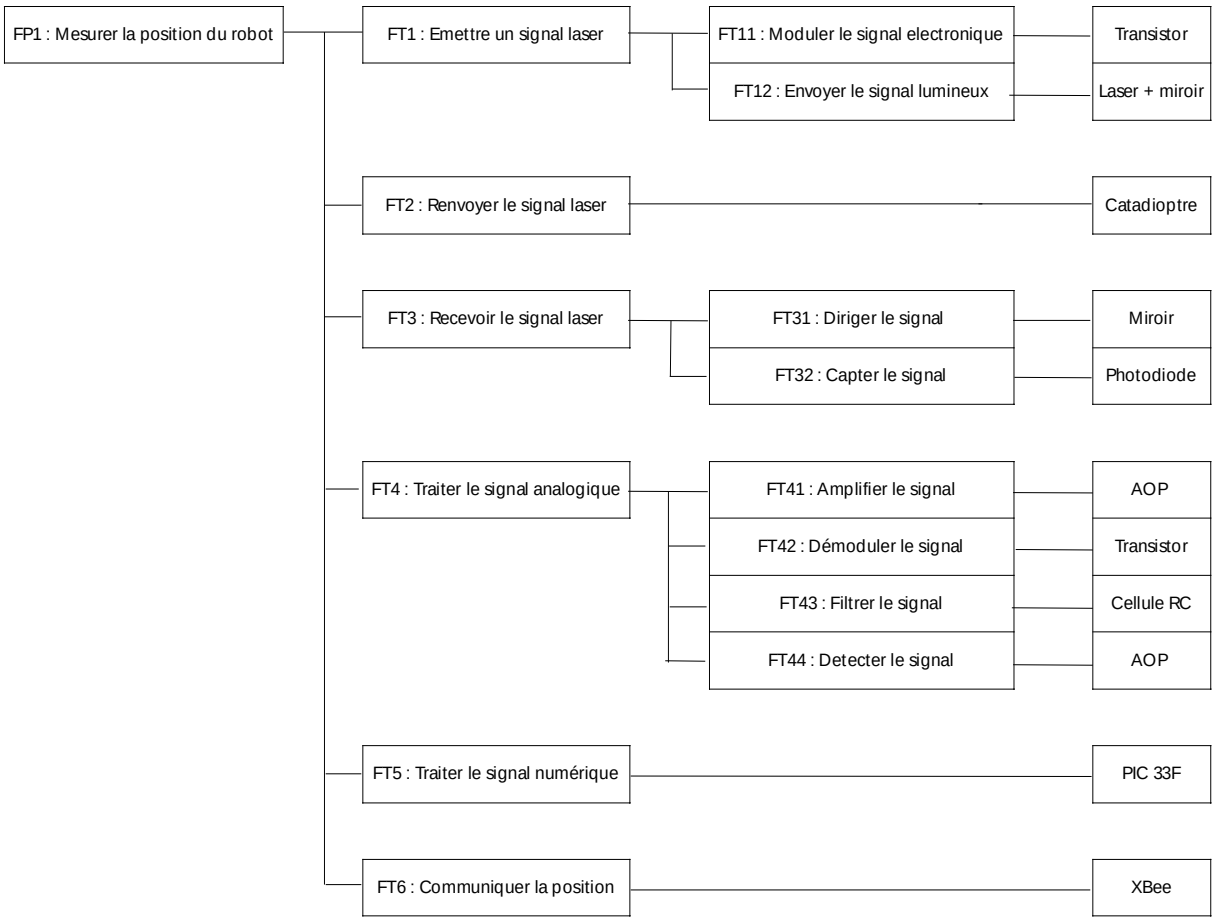
En effet la détection synchrone qui sera expliqué plus en détails un peu plus loin, nous permet en choisissant une fréquence f_{phelma} de ne traiter en réception que le signal de notre tourelle toujours dans le cas où l'adversaire opterait pour un système similaire.

Notre radar laser à détection synchrone nous permettra donc de mesurer les différents angles que forment la tourelle avec les balises extérieures comme on peut le voir sur la figure ci-dessous :

Le rayon laser sera réfléchit par les balises extérieurs à l'aide de catadioptres disposés sur celles-ci et ainsi le signal réfléchi sera réceptionné sur la tourelle en utilisant des photodiodes.

- **Diagramme Fonctionnel**

Ce diagramme permet de synthétiser la répartition des fonctions du projet. D'autre part, il fixe les solutions techniques associées à chaque fonction.



Performances des fonctions

FP1 : Nous cherchons à détecter le robot sur toute la table, ainsi nous verrons dans la suite du rapport la précision de détection par la méthode de la triangulation

FT11 : Limiter le bruit dû au transistor

FT12 : Envoyer le laser toujours à la même hauteur

FT2 : Renvoyer la lumière dans toutes les directions

FT32 : Recevoir une bonne partie du signal

FT41 : Limiter les bruits de l'AOP

FT42 : Limiter les bruits électroniques

FT43 : Filtrer de manière efficace tout ce qui ne contient aucune information

FT44 : Limiter les bruits électroniques

FT5 : Programmer le code de détection par triangulation

FT6 : Etablir une communication stable

IV. Description théorique de la réalisation :

IV.A) Théorie de la détection synchrone :

Soit V_i un signal continu et, par ailleurs un signal périodique X, de fréquence unique f_0. La modulation d'amplitude de X par V_i consiste simplement en la multiplication du second par le premier (connu sous le nom de 'porteuse') :

$$V(t) = V_i(t)X(t)$$

De même la démodulation consiste idéalement à multiplier le signal v par un signal de référence R, périodique de fréquence f_1, fréquence de la porteuse ; puis de filtrer le signal ainsi obtenu par un filtre passe-bas :

$$V_{out}(t) = V(t)R(t)$$

La détection synchrone consiste alors à démoduler le signal V(t) à l'aide de la porteuse ie telle que **X(t)=R(t)**

IV.B) Description mécanique du système :

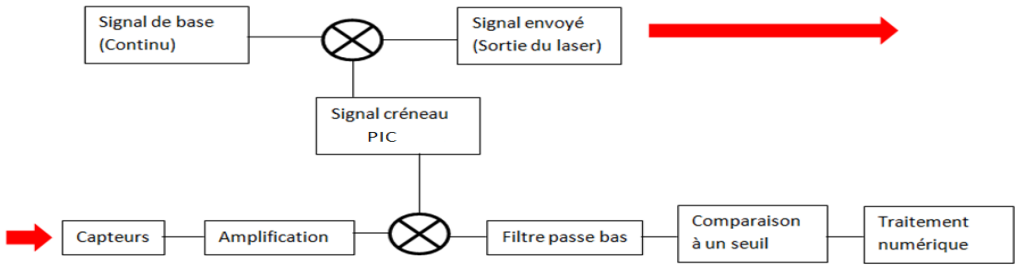
La description mécanique du système consiste ici dans la description de la tourelle de détection. En effet, comme expliqué dans la partie III/Solution retenue, nous souhaitons envoyer un rayon laser qui parcourrait le tour de la table et se réfléchissant sur les balises fixes pour enfin revenir au niveau de la tourelle.

La question est alors : **Comment faire en sorte que notre rayon laser fasse le tour de la table ?**

La solution choisie fût de disposer en face du laser un miroir tournant qui réfléchirait alors dans toutes les directions notre rayon laser et plus particulièrement en direction des balises fixes. Le rayon réfléchis serait alors réceptionné au niveau de la tourelle à l'aide de photodiodes. On pourra voir en Annexe le schéma pratique de la réalisation.

IV.C) Description électronique du système :

Pour pouvoir réaliser notre détection synchrone on réalise le schéma de principe suivant :



Chaque bloc joue un rôle dans la réalisation du système. Et nous allons en expliquer le rôle pour chacun :

1. La multiplication par le même signal en sortie du laser en en entrée du capteur a déjà été expliqué dans la partie théorique de la détection synchrone.

2. Le bloc amplificateur est indispensable avant tout traitement du signal. En effet, le signal en réception est fortement atténué à cause de la propagation dans l'air et du capteur.

3. Le filtre passe-bas de fréquence de coupure f_c nous permet de traiter le signal qu'autour de la fonction f_{phelma} et de supprimer une grande partie du bruit (lumière parasite, composants…)

4. La comparaison avec un seuil prédéterminé nous permet de savoir si la balise fixe a été effectivement touchée ou non.

5. Le traitement numérique lui nous donnera la position x,y du robot sur la table.

V. Réalisation du système :

V.A) Partie électronique :

V.A.1) Circuit détection synchrone :

- **Envoi du signal**

Comme vu précédemment dans la partie théorique, il faut envoyer un signal modulé afin de mettre en place ensuite une détection synchrone du retour de ce signal. De plus, il faut que le signal porteur soit de la même fréquence que le signal démodulant.

On a choisi d'envoyer un signal à partir d'un laser. Pour cela, la solution la plus simple est d'alimenter ce laser suivant un créneau d'une fréquence f choisie judicieusement afin de ne pas correspondre au bruit des lumières ambiantes (comme le 50Hz du secteur par exemple). Le signal créneau est émis par le PIC (plus facile à régler, plus précis et moins encombrant, vu que le circuit est déjà sur la plaque, qu'un vibrateur astable comme le NE555). On utilise ensuite un transistor en mode saturé

comme interrupteur pour faire passer la tension correcte et suffisamment de courant pour alimenter la diode laser (figure 2), car le PIC en lui-même n'est pas suffisant. On envoie ainsi un signal modulé par un créneau à …. kHz.

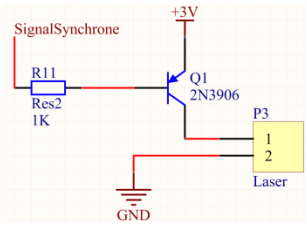


Figure 2 : Emission du signal

- **Réception du signal**

Pour récupérer le signal reçu, on utilise une photodiode qui s'apparente à un panneau solaire miniature (une petite plaque de silicium). On réalise autour de cette diode un circuit convertisseur courant tension avec un amplificateur opérationnel (figure 3). La capacité C7 sur le schéma permet de rendre l'amplification stable en compensant la capacité interne de la photodiode. N'ayant pas accès à la description technique de la photodiode utilisée, sa valeur a été déterminée expérimentalement. De même, la valeur de la résistance R15 est déterminée expérimentalement, afin de ne pas saturer l'amplificateur avec la lumière ambiante mais d'avoir tout de même un signal d'amplitude moyenne en sortie de l'amplificateur.

Directement après cet étage, nous avons prévu un amplificateur inverseur, si le signal n'est pas d'assez forte amplitude.

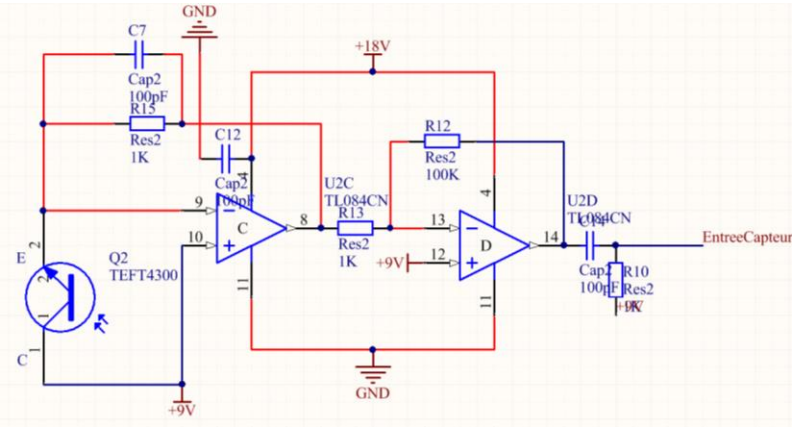


Figure 3 : Réception du signal

- **Démodulation du signal**

Afin de démoduler notre signal, nous devons multiplier le signal modulant issu du PIC et le signal obtenu à la sortie de l'amplification précédente. Cependant, il y a un détail à régler avant la multiplication elle-même : il faut gérer les composantes continues : afin de réaliser une détection synchrone, les signaux doivent être centrés à la masse (ici à 9V car l'alimentation n'est pas symétrique). Or le signal sortant du PIC est compris entre 0V et 3,3V, et le signal issu de

l'amplification est compris entre 9V et 18V : il faut donc supprimer les composantes continues gênantes, et centrer ces signaux en 9V : on utilise pour cela deux filtres RC.

Les signaux sont ensuite multipliés par le composant AD633 qui réalise l'opération suivante :

$$sortie = \frac{(entréeCapteur - 9) \times (9 - signalSynchrone)}{10}$$

Remarque : l'inversion de signalSynchrone par rapport à entréeCapteur est pour compenser l'amplificateur inverseur précédent (la détection synchrone ne fonctionne pas pour des signaux en opposition de phase)

Pour finir, le signal est filtré afin de ne retenir que la composante continue grâce à un filtre RC. Ce filtre a été dimensionné de sorte que la fréquence d'apparition du signal réfléchi (une dizaine de Hz) ne soit pas filtrée.

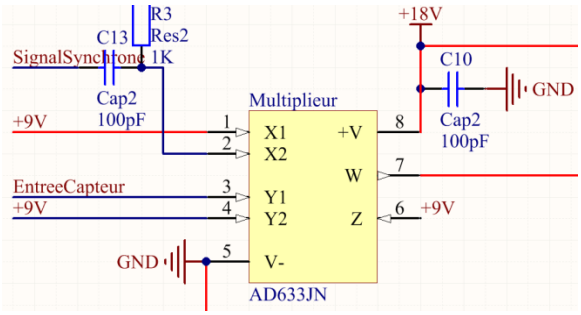


Figure 4 : Démodulation par multiplication

- **Préparation du signal pour le traitement numérique**

Avant d'envoyer le signal reçu au PIC pour traitement numérique, on l'amplifie une dernière fois (suiveur et amplificateur inverseur), puis on le compare avec une référence (réalisée avec un potentiomètre et deux résistances) grâce à un LM311 afin de passer le signal en numérique avec une amplitude de 3,3V requise par le PIC

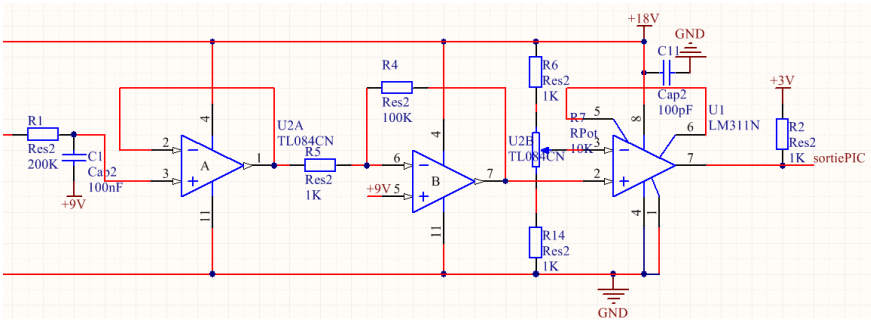


Figure 5 : Filtra passe-bas et préparation au traitement numérique

V.A.2) Circuit PIC :

Afin de faire fonctionner le PIC et de le programmer, un circuit autour de celui-ci est nécessaire. Ce circuit nous a été fourni par la datasheet du PICKit (programmateur du PIC), il est représenté figure 5 :

Les connections ont été choisies en fonction des capacités du PIC (PWM, i/o, uart) et une led de débogage a été ajoutée.

Le choix de ce microcontrôleur peut paraitre étrange : il y a beaucoup trop d'entrée sortie, de mémoire : il est surdimensionné. Cependant, pour le projet, il a été choisi car nous avions des notions de programmation sur ce PIC-là et des codes d'exemples pour les PWM et l'uart qui était accessible facilement. De plus, ce PIC possède plusieurs PWM matériels, ce qui est pratique pour ce que nous avons à faire (moteur, signal à 40kHz).

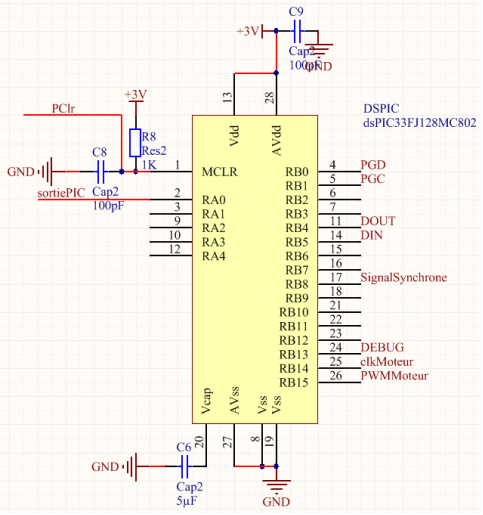


Figure 6 : Circuit de programmation du PIC et différentes connections

V.A.3) Moteur :

Pour le projet, compte tenu des fortes contraintes d'espace, il nous fallait un moteur petit et fin. De plus, il devait être facilement contrôlable, avec de faibles vibrations, tournant assez vite et avec un couple quelconque. C'est pour cela que nous avons opté pour un moteur brusheless. Au départ, l'objectif était de faire tourner un moteur de disque dur, puis nous avons trouvé un moteur de ce type avec un driver intégré, ce qui nous a facilité la tâche. Après quelques tests de connexion et recherches internet sur le driver, nous avons pu déterminer l'utilité de chaque connexion et ainsi envoyer les signaux pour faire tourner le moteur correctement :

- Une tension d'entrée continue d'au moins 15V
- Une horloge
- Un PWM qui permet de réguler la vitesse.

Une amélioration à apporter serai de récupérer le signal issu des capteurs à effet hall sous le moteur, afin de savoir exactement la vitesse et d'asservir correctement le moteur grâce au PIC (en effet, la

vitesse du moteur dépend non seulement du PWM, mais aussi de la tension d'alimentation)

Figure 7 : Moteur utilisé

V.A.4) Alimentation :

Etant donné l'espace réduit dont on dispose, nous allons réunir tous les composants sur un même circuit, ce qui pose plusieurs problèmes.

Tout d'abord le problème le plus évident : les différentes alimentations nécessaires :

- 3,3V continu pour le PIC
- 5V continu pour le module XBee
- Minimum 15V pour le moteur
- 3,3V pour le laser
- Une tension symétrique pour les amplifications, comparaison, multiplication

De plus, les masses du PIC et du moteur doivent correspondre pour que le moteur fonctionne (même masse entre l'alimentation et les signaux de contrôle)

Afin de résoudre ces problèmes, nous avons opté pour une alimentation par deux piles 9V, ainsi que deux régulateurs (3,3V et 5V). Cette solution, la plus simple et la moins couteuse, est loin d'être parfaite pour ces différentes raisons :

- La tension varie aux bornes des piles en fonction de leur usure et de leur utilisation (de 9V à 7V): cela affecte la vitesse du moteur et les opérations analogiques sur le signal (l'alimentation n'est plus exactement symétrique). Solution : utiliser une batterie plus performante et avec une chute de tension moins importante.
- La régulation utilisée est une régulation avec dissipation par effet joule : elle chauffe énormément dès qu'on utilise le laser et le PIC. La mise en place d'un radiateur est obligatoire (perte d'espace), et la consommation est trop importante pour une balise qui est censé être nomade. La solution serait d'utiliser une régulation plus adaptée, comme une alimentation à découpage. Celle-ci est cependant plus chère.

Un autre problème est la cohabitation analogique/numérique/puissance : l'alimentation est fortement bruitée. Pour résoudre en partie cela, on a placé une capacité de filtrage à chaque alimentation de circuit. La solution idéale serait d'avoir des alimentations différentes, mais ceci est impossible avec des piles 9V (contrainte de taille de la balise). Une solution autre reste donc à trouver.

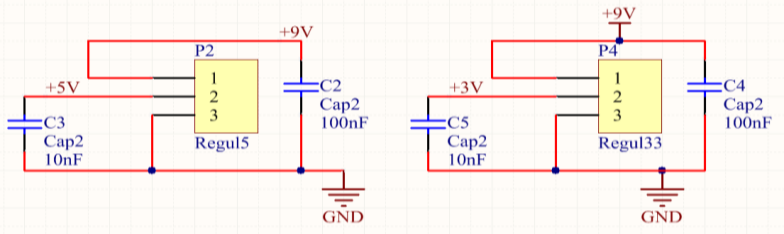
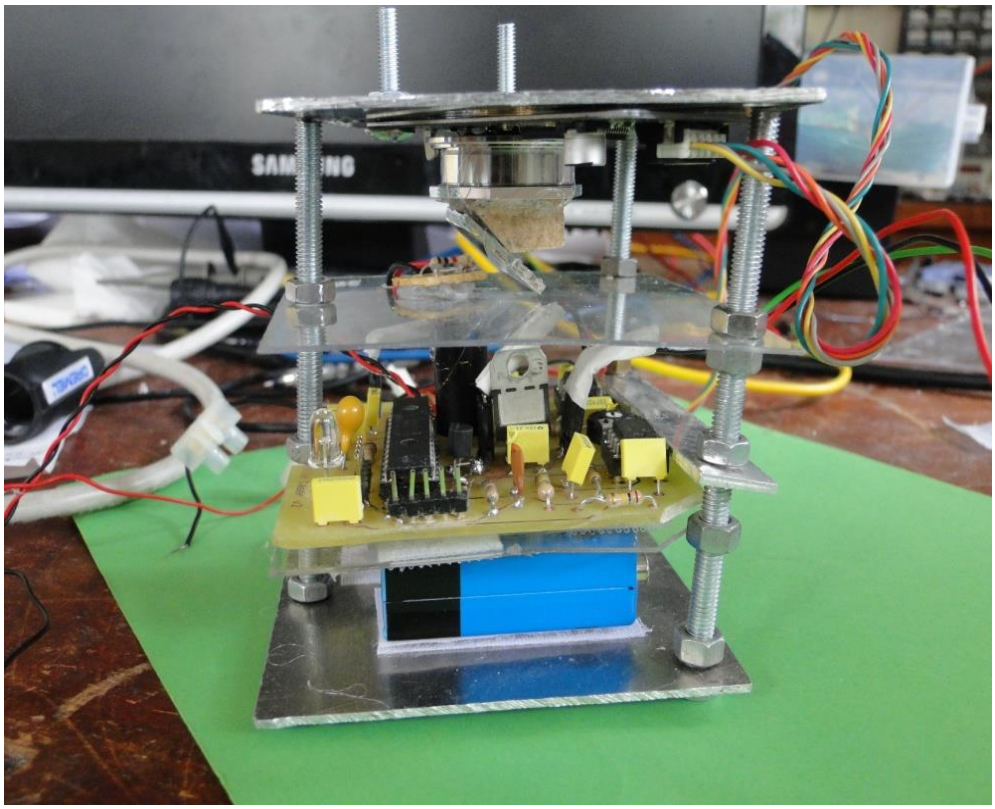


Figure 8 :Cablage des deux régulateurs

V.B) Partie Mécanique :

La réalisation mécanique de la tourelle s'est faite au moyen de 2 plaques d'aluminium (8x8cm) de 3 vis de hauteur 9cm environ, de boulons et de Velcro. On y a incorporé comme prévu le circuit électronique, les 2 piles 9v, le moteur, le miroir ainsi que le laser. On a ainsi obtenue la tourelle suivante :

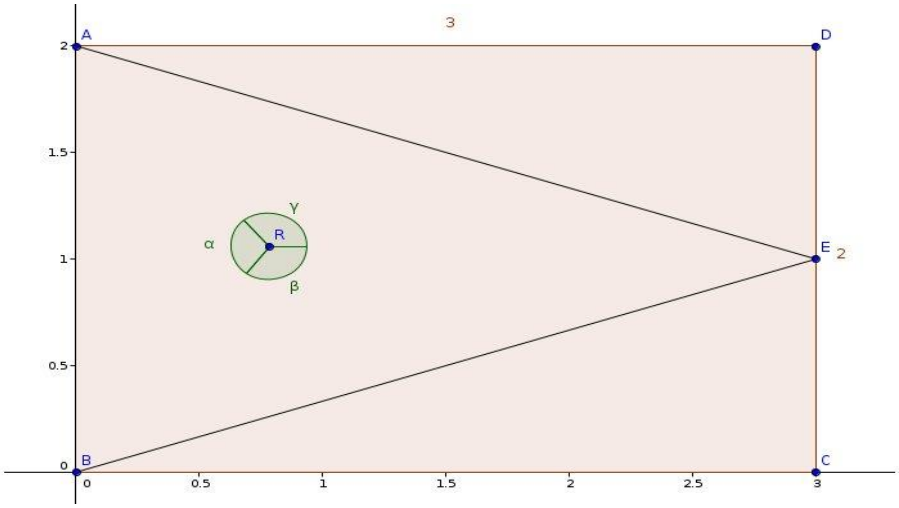
On pourra trouver en annexe du rapport, des photos légendés de la tourelle.



V.C) Traitement Numérique :

L'objectif de cette partie est de calculer la position du robot en fonction des temps relevés entre les balises. A partir des temps, on détermine la valeur des angles (α, β, γ) puis par des calculs trigonométriques, on détermine la position cartésienne du robot.

V.C.1) Fonctionnement du code source :



La table de jeu

Le code source a pour objectif de calculer la position du robot à partir des intervalles de temps entre les balises. Afin de différencier les différentes balises, la balise située au point A est double (la réflexion sur le catadioptre est divisée par une bande de scotch).

Voici les prototypes des fonctions utilisées :

```
void calculPi(int gamma, double coordonnee[]);
void calculDiffPi(int alpha, int gamma, double coordonnee[]);
```

Ces deux fonctions sont la traduction en langage C du paragraphe V.C.2), calculant la position (x, y) en fonction de l'angle α et γ selon si α est égale ou non à π.

```
int calculAngle(int t[], int angle[], int somme);
```

Cette fonction calcule les angles α et γ en fonction des intervalles de temps entre les balises.

```
double mesureTemps(int t[]);
```

Celle-ci relève les temps entre les balises elle sera détaillé dans la partie V.C.3).

```
void envoie(double pos[]);
```

Cette dernière envoie les résultats au robot cette partie n'a pas été développée.
Pour diminuer la complexité des calculs effectués par le microcontrôleur nous avons utilisé des tableaux de valeurs pour les fonctions trigonométriques.

```
static double cos[361], sin[361], tan[361];

void initCos(double cos[]);
```

```
void initSin(double sin[]);
void initTan(double tan[]);

void initCos(double cos[])
{
        cos[0] = …
```

Ces fonctions étant longues et sans intérêt particulier elles ne seront pas détaillés ici.

Description de la fonction principale (main) :

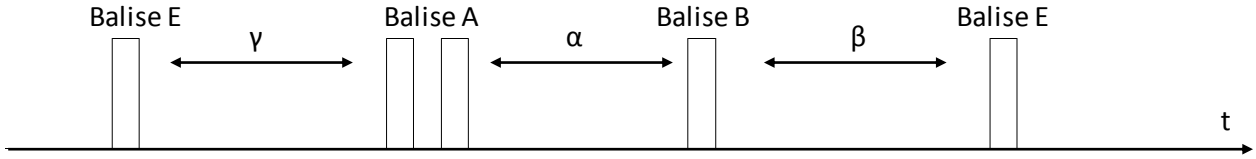
Dans un premier temps on va chercher à connaitre la durée d'un tour pour pouvoir contrôler qu'une balise n'est pas été manquée par notre système par exemple si une tige de l'armature est alignée avec une balise. Pour cela on réalise 20 tours et l'on retient le plus petit temps mis pour croiser les 5 balises : 3 balises différentes dont une double et pour finir un tour on revient sur la première balise.

```
/* INITIALISATION */
for (i = 0; i < 20; i++)
{
        somme = mesureTemps(t);
        if (somme < dureeTour || dureeTour == 0)
                dureeTour = somme;
}
```

Ensuite en fonctionnement normal, on va relever les différents intervalles de temps, on contrôle si on n'a pas manqué une balise en comparant avec la durée calculés à l'initialisation, si la durée est correcte on fait les calculs ensuite on envoie les résultats puis on reboucle. On définit une constante d'erreur à adapter selon la durée d'un tour.

```
#define ERREUR …

/* MARCHE NORMAL */
while ()
{
        somme = mesureTemps(t);
        if (somme > dureeTour - ERREUR && somme < dureeTour + ERREUR)
        {
                calculAngle(t, angle, somme);
                if (angle[0] > 179 && angle[0] < 181)
                        calculPi(angle[1], pos);
                else
                        calculDiffPi(angle[0], angle[1], pos);
                envoie(pos) ;
        }
}
```

Description de la fonction calculAngle :



On récupère les 4 intervalles de temps (entre chaque balises plus un pour la balise double), on commence par rechercher le plus petit qui correspondra à la balise double et ensuite selon son numéro dans le tableau on en déduit α et γ (β ne sert pas pour les calculs), l'intervalle entre la balise double et la suivante représente l'angle α, ensuite vient l'angle β et enfin l'angle γ.

```
int calculAngle(int t[], int angle[], int somme)
{
        //Recherche du minimum
        int  i, j = 0;
        for (i = 1; i < 4; i++)
                if (t[i] < t[j])
                        j = i;
        //Distinction des cas selon l'indice du minimum
        switch (j)
        {
                case 0 :
                        angle[0] = 360*t[1]/somme;
                        angle[1] = 360*t[3]/somme;
                 break;
                 case 1 :
                        angle[0] = 360*t[2]/somme;
                        angle[1] = 360*t[0]/somme;
                 break;
                 case 2 :
                        angle[0] = 360*t[3]/somme;
                        angle[1] = 360*t[1]/somme;
                 break;
                 case 3 :
                        angle[0] = 360*t[0]/somme;
                        angle[1] = 360*t[2]/somme;
                 break;
        }

        return 0;
}
```

V.C.2) Calcul de la position par triangulation :

Dans cette partie on considère les angles (α, β, γ) connues, on cherche les coordonnées cartésiennes du point R.

D'abord nous allons définir quelques constantes pour la résolution du problème.

On note AE = BE = d, AB = CD = l, AD = BC = L et les angles seront notés $\widehat{EAB}$: l'angle entre AE et AB, $\widehat{EAD} = \widehat{EBC} = \theta, \widehat{ARB} = \alpha, \widehat{BRE} = \beta, \widehat{ARE} = \gamma$.

On applique le théorème du sinus :

$\frac{\sin(\widehat{BAR})}{BR} = \frac{\sin(\widehat{ABR})}{AR} = \frac{\sin\alpha}{l}$ (1)

$\frac{\sin(\widehat{EAR})}{ER} = \frac{\sin(\widehat{AER})}{AR} = \frac{\sin\gamma}{d}$ (2)

$\frac{\sin(\widehat{BER})}{BR} = \frac{\sin(\widehat{EBR})}{ER} = \frac{\sin\beta}{d}$ (3)

De l'équation (1) et (3) on en déduit :

$l * \sin(\widehat{ABR}) = AR * \sin\alpha$ et $AR = \frac{d*\sin(\widehat{AER})}{\sin\gamma}$ ($\gamma \neq 0$ car $\gamma \in [90, 270]$)

$\rightarrow l * \sin(\widehat{ABR}) * \sin\gamma = d * \sin(\widehat{AER}) * \sin\alpha$

Or $\widehat{ABR} = 180 - \alpha - \widehat{BAR}$, $\widehat{AER} = 180 - \gamma - \widehat{EAR}$ et $\widehat{EAR} = 90 - \theta - \widehat{BAR}$

$\rightarrow l * \sin(180 - \alpha - \widehat{BAR}) * \sin\gamma = d * \sin(90 - \gamma + \theta + \widehat{BAR}) * \sin\alpha$

$\rightarrow l * \sin(\alpha + \widehat{BAR}) * \sin\gamma = d * \cos(-\gamma + \theta + \widehat{BAR}) * \sin\alpha$

$\rightarrow l * (\sin\alpha * \cos\widehat{BAR} + \cos\alpha * \sin\widehat{BAR}) * \sin\gamma$
$= d * (\cos(\theta - \gamma) * \cos\widehat{BAR} - \sin(\theta - \gamma) * \sin\widehat{BAR}) * \sin\alpha$

$\rightarrow l * (\sin\alpha + \tan\widehat{BAR} * \cos\alpha) * \sin\gamma = d * (\cos(\theta - \gamma) - \sin(\theta - \gamma) * \tan\widehat{BAR}) * \sin\alpha$

$$\tan\widehat{BAR} = \sin\alpha * \frac{d * \cos(\theta - \gamma) - l * \sin\gamma}{d * \sin(\theta - \gamma) + l * \tan\alpha * \sin\gamma}$$

$X = BR * \sin\widehat{ABR} = BR * \sin(180 - \alpha - \widehat{BAR}) = BR * \sin(\alpha + \widehat{BAR})$

$= BR * (\cos\alpha * \sin\widehat{BAR} + \sin\alpha * \cos\widehat{BAR})$

$Y = BR * \cos\widehat{ABR} = BR * \cos(180 - \alpha - \widehat{BAR}) = -BR * \cos(\alpha + \widehat{BAR})$

$= BR * (\sin\alpha * \sin\widehat{BAR} - \cos\alpha * \cos\widehat{BAR})$

et d'après (1), $BR = l * \frac{\sin(\widehat{BAR})}{\sin\alpha}$ et si $\alpha \neq \pi$

$$X = l * \sin^2(\widehat{BAR}) * (\frac{1}{\tan\alpha} + \frac{1}{\tan\widehat{BAR}})$$

$$Y = l * \sin^2(\widehat{BAR}) * (1 - \frac{1}{\tan\alpha} * \frac{1}{\tan\widehat{BAR}})$$

$$\sin^2\widehat{BAR} = 1 - \frac{1}{1 + \tan^2\widehat{BAR}}$$

si α = π,

$$X = 0$$

$$Y = \frac{l}{2} - \frac{L}{\tan \gamma}$$

V.C.3) Programmation spécifique au PIC :

Dans ce projet, nous avons besoin, en plus du programme qui permet d'analyser les résultats reçu, de paramétrer les différents périphériques du microcontrôleur : entrées/sorties, PWM et timer. C'est ce qu'on va vous exposer dans cette partie.

- **Entrées/sortie**

Dans ce projet, nous avons besoin d'une entrée qui est le résultat de l'acquisition par démodulation synchrone et d'une sortie qui correspond à la led de débogage. L'orientation entrée/sortie se fait dans le registre TRIS (0 = sortie, 1= entrée) :

```
//Mise en sortie de la led débogage et en entrée du signal reçu
TRISBbits.TRISB13 = 0 ;
TRISAbits.TRISA0 = 1 ;
```

On écrit ensuite les valeurs de sorties pour la led dans le registre LAT (1 = état haut, 0=état bas) :

```
//Allumage de la led
LATBbits.LATB13 = 0 ;
//Extinction de la led
LATBbits.LATB13 = 1 ;
```

On peut lire les valeurs d'entrée dans le registre PORT :

```
if(PORTAbits.PORTA0 == 1)
{
      //on a reçu un signal
}
```

Voilà qui conclue la partie sur les entrées/sorties nécessaire à notre projet.

- **Timer**

Afin de pouvoir paramétrer le timer, il faut auparavant avoir paramétré l'horloge principale du PIC. Cette partie n'a pas été réalisée dans le cadre de ce projet, nous avons utilisé un code d'exemple : On supposera que l'horloge principale est à 80 MHz.

```
void initTimer()
```

```
{
        TMR1 = 0x0000; //Met le timer à 0
        PR1 = 64000; //Maximum du timer
        IFS0bits.T1IF = 0;
        IEC0bits.T1IE = 0;
        T1CONbits.TCKPS=0b11; //on divise par 256 la fréquence d'horloge
}

//remettre le timer à zero après une acquisition :
TMR1 = 0 ;
//récupérer la valeur actuelle du timer :
Valeur = TMR1 ;
//lancer le timer (une fois au début) :
T1CONbits.TON = 1 ;
```

Remarque : Pourquoi diviser par 256 la fréquence d'horloge ? On a une fréquence d'horloge interne de 80MHz, d'où une fréquence de cycle de 40MHz. Si on prend comme hypothèse que le moteur fait un tour en 0,3s, et qu'on veut une résolution de l'ordre du degré (360 par tour), il nous faut au minimum une incrémentation du timer toute les 0.008s environ, c'est-à-dire à une fréquence de 1,2 kHz. Or on ne peut pas diviser la fréquence simplement (avec un seul timer) plus de 256 fois. Cependant cela suffit, car si on fait bien des tours à environ 3Hz, on arrivera à 47000 au compteur du timer à la fin du tour, ce qui est inférieur à 64000 (environ la taille maximum d'une variable de type `int` dans ce microcontrôleur) (on arrive bien à compter pendant un tour).

- **Implantation**

On initialise le timer puis on relève 5 fois la valeur du timer (à chaque front montant de l'entrée) puis en les soustrayant de proche en proche on obtient 4 intervalles de temps consécutifs correspondant à l'intervalle en chaque balise (deux balises simple et une balise double).

```
double mesureTemps(int t[])
{
        int i;
        int valeurTimer[5];

        TMR1 = 0 ; // on reinitialise le timer

        for (i = 0; i < 5; i++)
        {
                // attente front montant
                while (PORTAbits.PORTA0);
                while (!PORTAbits.PORTA0);

                valeurTimer[i] = TMR1; // on recupere la valeur du timer
        }
        for (i = 0; i < 4; i++)
                t[3 - i] = valeurTimer[4 - i] - valeurTimer[4 - (i + 1)];

        return t[0] + t[1] + t[2] + t[3];
}
```

- **PWM**

Comme précédemment, l'horloge interne doit être paramétrée avant de paramétrer le PWM. Voilà le code d'initialisation :

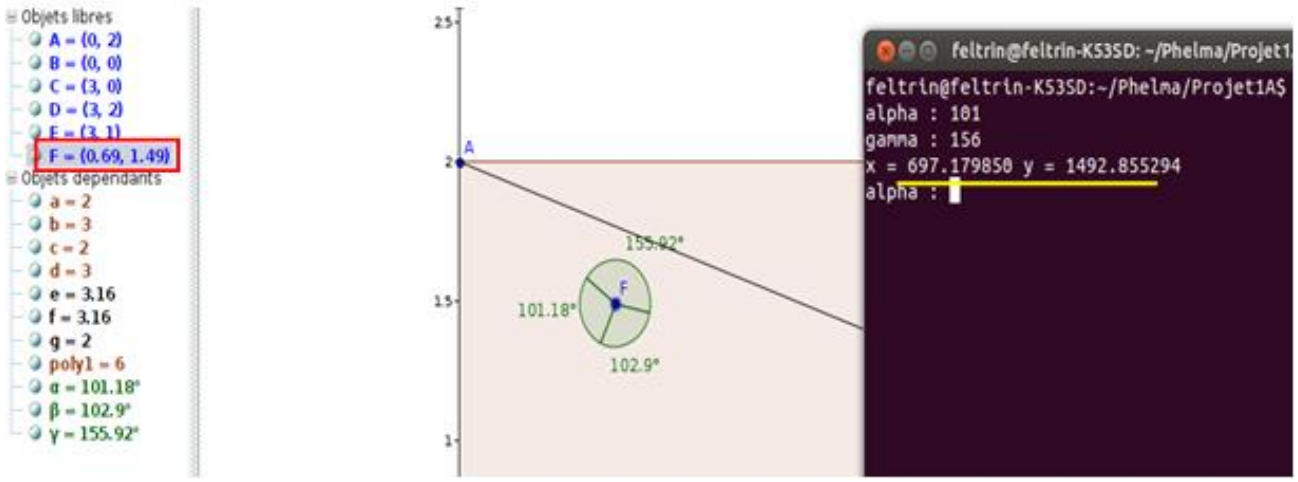
```
void initPWM()
{
    P1TCONbits.PTEN = 1; //Activation de la base de temps du PWM
    P1TCONbits.PTCKPS = 0b11; //On divise par 64 la fréquence de cycle
    P1TPER = 2000; //période d'un PWM

//Activation des PWM nécessaires au projet
    PWM1CON1bits.PMOD1 = 1;
    PWM1CON1bits.PMOD2 = 1;
    PWM1CON1bits.PEN1H = 1;
    PWM1CON1bits.PEN1L = 1;
    PWM1CON1bits.PEN2H = 1;
    PWM1CON1bits.PEN2L = 1;

    P1DC1 = 1000; //Temps passé en haut pour le premier PWM (50% : c'est
notre horloge)
    P1DC2 = 500; //Temps passé en haut pour le deuxième PWM (25% : moteur)
}
```

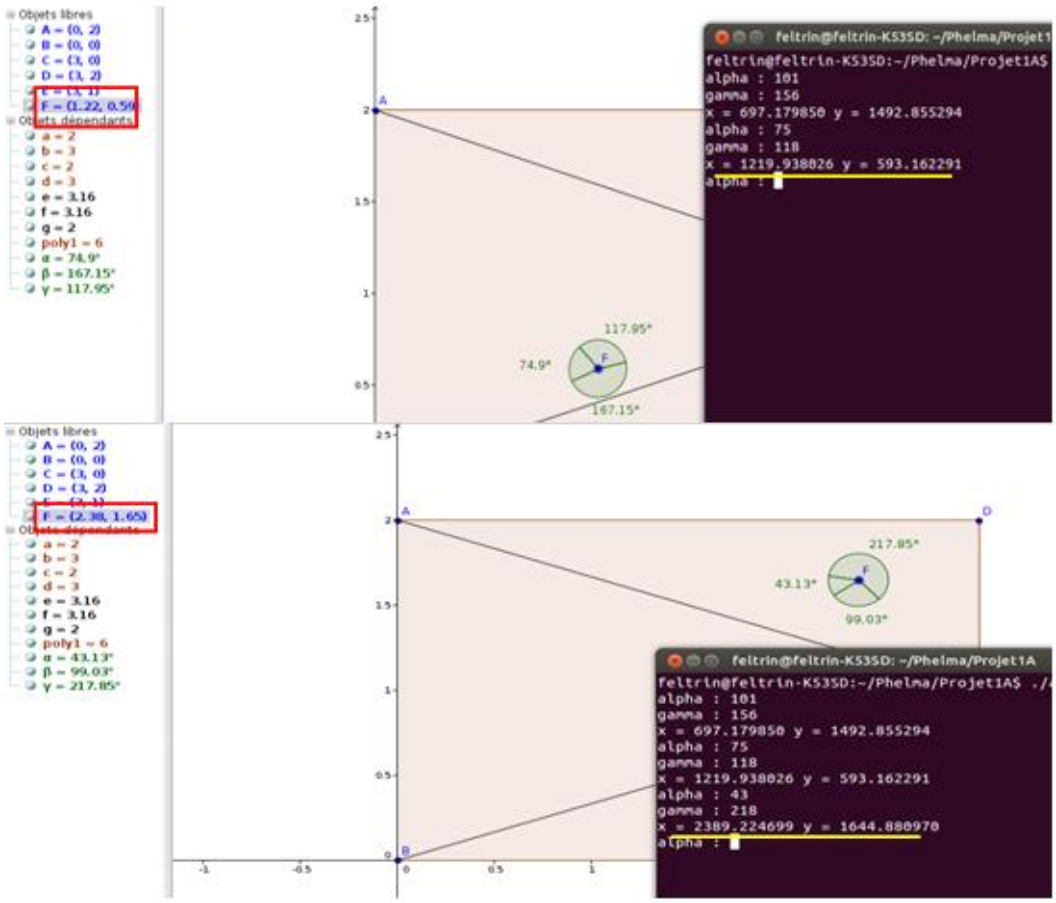
Remarque : P1DC2 est réglable ailleurs dans le programme, on peut modifier cette valeur pour asservir le moteur à une vitesse quasi constante.

V.C.4) Tests du programme :

Afin de vérifier que nos calculs étaient correctes et que notre programme fonctionnait nous avons utilisé la logiciel GeoGebra, logiciel qui nous a permis de représenter la table de jeux et de déplacer le point représentant le robot sur le terrain. Ainsi à partir des angles d'une position quelconque nous avons pu déterminer la position cartésienne du robot.

Voici quelques exemples de tests :





VI. Résultats et difficultés rencontrées :

VI.A) Partie électronique :

Nous avons dans un premier temps réalisé le montage sur plaque à trous afin de tester les différents « étages » de notre montage : réception, amplification, multiplication, préparation du signal pour le traitement numérique.

Il y a eu plusieurs étapes, et le circuit n'a pas toujours été tel qu'on vous l'a présenté ci-dessus. Nous avions commencé par travailler sur un filtre passe bas plus complexe, afin de mieux sélectionner la composante continue à la sortie du multiplieur (filtre à variable d'état). Comme nous l'a fait remarquer notre tuteur à ce moment-là du projet, un tel filtre est inutile dans notre circuit, et encombrant (beaucoup d'amplificateur opérationnels). De plus, avant d'utiliser un AD633 comme multiplieur afin d'effectuer la démodulation, nous pensions simplement utiliser un transistor comme interrupteur pour « multiplier » par un créneau compris entre 0V (interrupteur fermé) et 1V (interrupteur ouvert). C'était sans compter sur ce qui sera durant tout le projet notre principale difficulté : les composantes continues. Nous avons réussi à réduire leur impact en utilisant un multiplieur AD633 et en plaçant un filtre passe haut à chaque entrée de celui-ci.

Ci-dessous, vous trouverez quelques oscillogrammes illustrant les premières étapes du traitement du signal reçu :

Les figures 9 et 10 montrent la sortie du convertisseur courant tension. La figure 6 est réalisée sans condensateur en parallèle de la résistance, alors que alors de l'acquisition de la figure 7, une capacité de 10 pF était présente en parallèle de la résistance : on arrive dans les mêmes conditions d'expérience à gagner 0,65V, ce qui n'est pas négligeable (gain de 33% de tension).

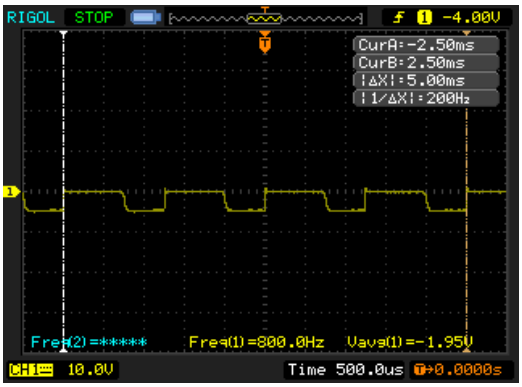


Figure 9

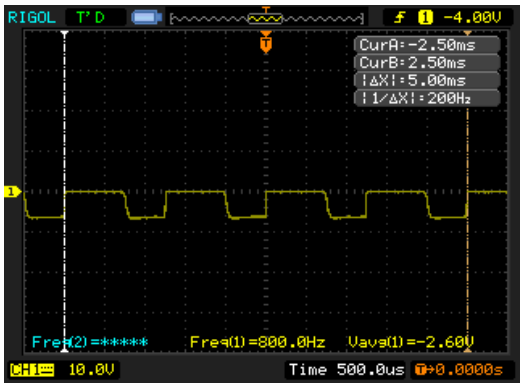


Figure 10

La figure 11 montre la sortie du multiplieur. On remarquera que tous les problèmes de composantes continus ne sont pas réglés : on devrait avoir un signal centré en 0V, hors on est plus aux alentours de -10V.

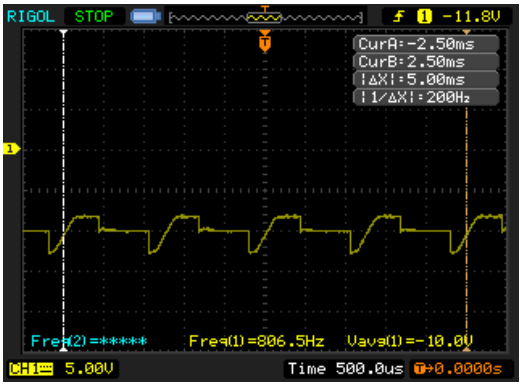


Figure 11

VI.B) Traitement numérique :

Pour alléger notre programme nous avons approximé les angles à des entiers ce qui nous permet de travailler avec des type de variable plus léger pour le microcontrôleur, le type int. Bien sûr cette approximation se répercute sur notre précision dans la localisation du robot, nous avons donc cherché à estimer cette résolution pour voir si considérer les angles entiers ne nous empêchaient pas définir la position du robot partout sur la table.

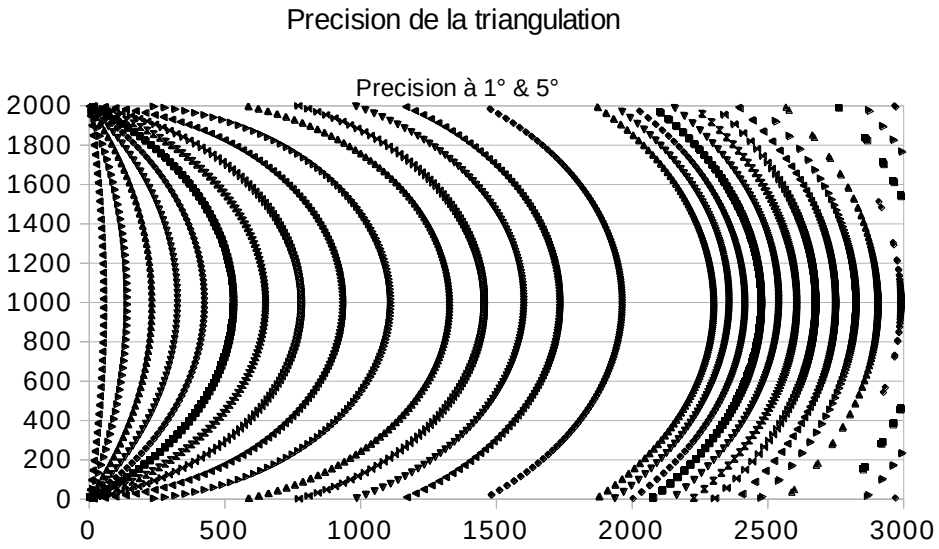


Figure 12 : Positions calculées avec une précision de 5° sur la première moitié de la table & précision de 1° sur la seconde moitié

- **<u>Comment les positions ont-elles été calculées ?</u>**

Ce schéma représente la vraie table de jeu. La dimension est de 3000mm*2000mm. Pour pouvoir avoir une idée des positions exactes que l'on détecte, nous avons utilisé un tableur qui calcule grâce au système d'équations de triangulation les positions.

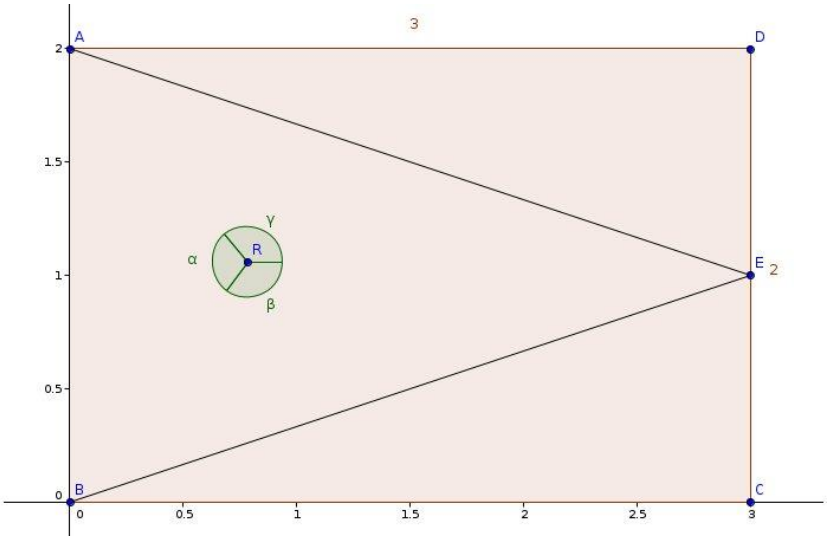


Figure 13 : La table de jeu

Plus précisément, le point de la figure 2 est déterminé grâce à 3 angles alpha beta et gamma qui sont définis par la position des balises extérieures.

Pour comprendre comment le schéma figure 1 a été tracé, il faut exhiber **2 conditions sur les angles**

- Chaque angle à une plage de variation qui nous intéresse, c'est-à-dire pour que le point reste dans la zone ([0,3000];[0,2000]).

- $\alpha + \beta + \gamma = 360°$

Donc on se fixe α par exemple égal à 90° et on cherche l'ensemble des angles β et γ qui vérifient ces 2 conditions. Une fois cet ensemble trouvé, on le réinjecte dans le système d'équations de triangulation et on trace les positions.

L'étape la plus délicate de ce procédé est d'exhiber l'ensemble de 2 angles qui vérifient les 2 conditions c'est pour cela que nous avons travaillé par élimination de cas.

On a tracé tous les points possibles avec $\alpha = 90°$ et puis on élimine par l'utilisation de fonctions « SI » les positions incohérentes. (>3000 ou <0 par exemple)

Grâce à un logiciel de géométrie, on a pu connaître approximativement les ensembles de définition des 3 angles, ce qui nous a permis d'avancer plus efficacement dans la recherche des positions du robot.

- **Que peut-on en déduire ?**

La complexité des calculs rendait très difficile la superposition des différents tracés pour chaque angle. C'est pourquoi nous avons décidé de montrer la différence des précisions à 5° et 1° sur la même table de jeu (la zone blanche au milieu est pour séparer)

Premièrement, on remarque que lorsque l'on fait varier notre alpha de 1°, ici on voit sur la droite du schéma la variation de $\alpha = 34° \rightarrow \alpha = 47°$, la précision est **excellente**, on détecte le robot à 5cm près ce qui est largement suffisant au vu de la taille de ce dernier.

Deuxièmement, on note aussi le manque de précision dans les 2 coins proches de la balise isolée de droite, intuitivement ce résultat est cohérent une seule balise ne permet pas d'avoir au tant d'informations que les 2 balises de gauche où il n'y a aucune zone « floue ». Ainsi, il faudra tenir compte de ce défaut dans **la stratégie de match** pour ne pas se faire éliminer...

Enfin, d'après les résultats sur la partie gauche de la table on conclut que si notre système à une résolution de +- 5°, il aura du mal à détecter et garder en mémoire la position du robot en mouvement.

VII. Gestion de projet :

Afin de gérer au mieux le projet, nous avons en début de semestre définit des responsabilités à chacun :

- **Chef de projet :** Adrien THIBEAUD--rôle : Coordonner le travail et vérifier l'avancement du projet
-Trésorier : Eric FELTRIN--rôle : Définir le budget, gérer les documents comptable et les commandes
- **Secrétaire générale :** Sahbi THAIRI--rôle : Faire le bilan des séances et préparer les rapports
- Responsable de communication : Aziz BELHASSAN--rôle : Prendre des photos et/ou vidéos du projet et superviser la réalisation du support de communication

En plus de ces différentes responsabilités, nous avons réaliser un diagramme de GANTT (voir Annexe) qui nous permettra de gérer au mieux le temps à notre disposition. Le diagramme de GANTT reprend ainsi les différentes tâches à réaliser et l'agenda associé.

Le diagramme est bien partagé en plusieurs parties, afin que chacun puisse travailler en parallèle, car les tâches ne sont pas toutes dépendantes. Le choix de commencer par ce concentrer sur l'électronique n'est pas anodin : en effet, il est nécessaire d'avoir le circuit pour teste à la fois la position des capteurs et laser dans le prototype, mais aussi le programme informatique bien que celui-ci puisse être testé dans un premier temps grâce à des signaux de test (GBF).

Nous avons également définit des critères concernant l'évaluation latérale en leur affectant des coefficients :

Critère d'évaluation	Coefficient affecté
Ponctualité	1.5
Travail hors séances	1
Travail fourni durant les séances	2.5
Rôle tenu	1.5
Communication/cohésion	2

On trouvera alors en Annexe, les notes obtenues lors de cette évaluation

VIII. Conclusion :

Ce projet fut une enrichissante expérience, en effet nous avons du apprendre à mener à bien la gestion et la réalisation technique d'un projet complexe. Il a permit de mettre en œuvre nos connaissances théoriques acquises durant l'année, mais aussi de rendre compte des compétences de gestion nécessaire à l'ingénieur.

Points forts

- Mise en place des responsabilités de chacun
- Bonne gestion du retard relatif au Gantt
- Communication entre les membres du groupe

Si c'était à refaire

- Tester la partie électronique plus rapidement
- Pas encore de test complet du système avec 3 balises

On note d'ailleurs des différences dans les résultats des parties informatiques, mécaniques et électroniques. En effet, la partie informatique offre les résultats attendus tandis que la partie électronique sujette à plus d'instabilité ne fonctionne pas comme attendue. En effet, le microcontrôleur PIC est instable depuis quelque temps et perturbe le déroulement de nos tests, car il est la source, entre autres, du signal synchrone.

ANNEXE

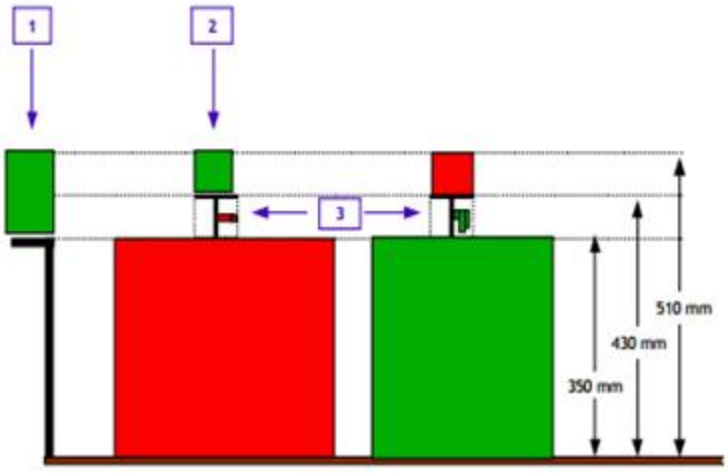


Figure 1

Légende :

- 1 : Balise fixe (dimensions maximales L x l x h: 80 x 80 x 160 mm)
- 2 : Balise embarquée (dimensions maximales L x l x h: 80 x 80 x 80 mm
- 3 : Mât du support (pouvant accueillir des capteurs et éléments associés uniquement, à condition de rester à l'intérieur de la projection verticale du support de balise)

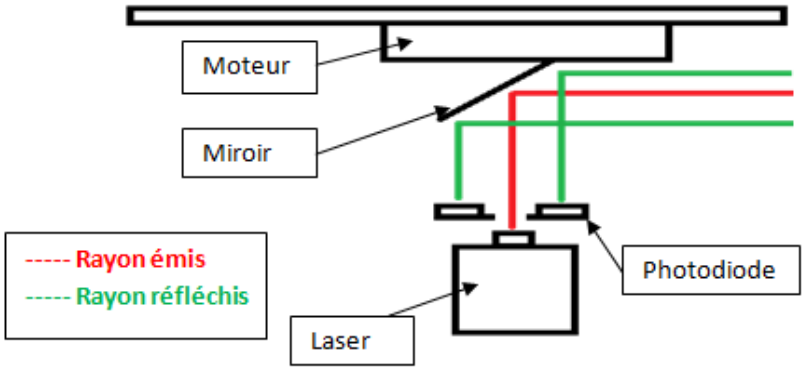


Figure 2

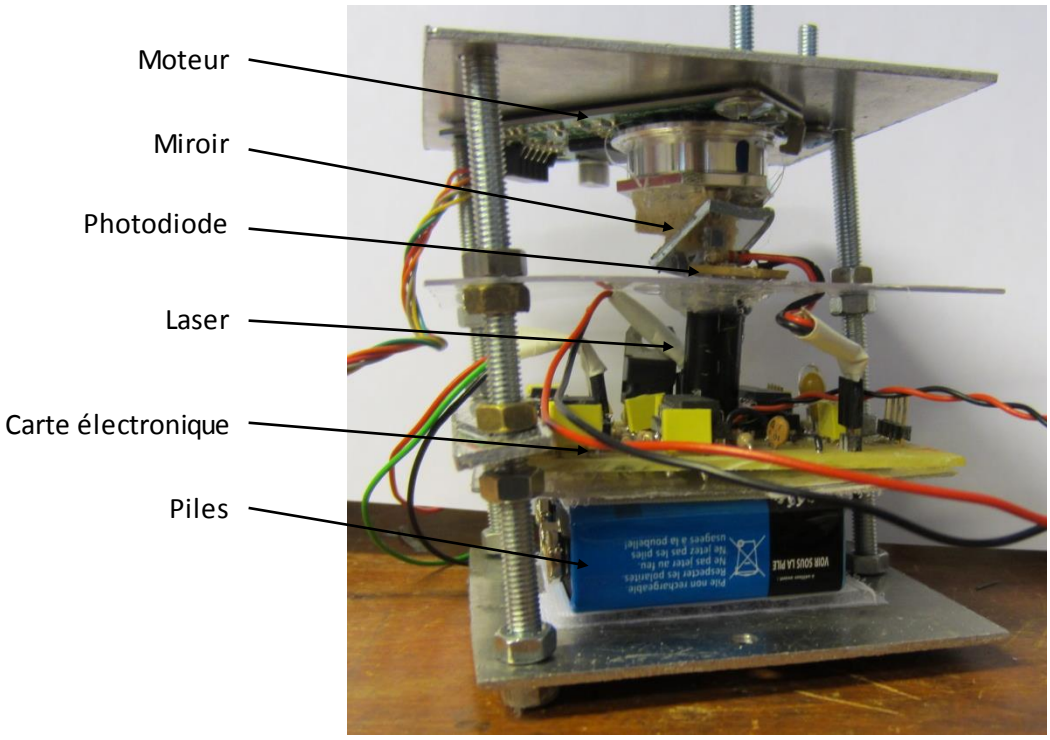
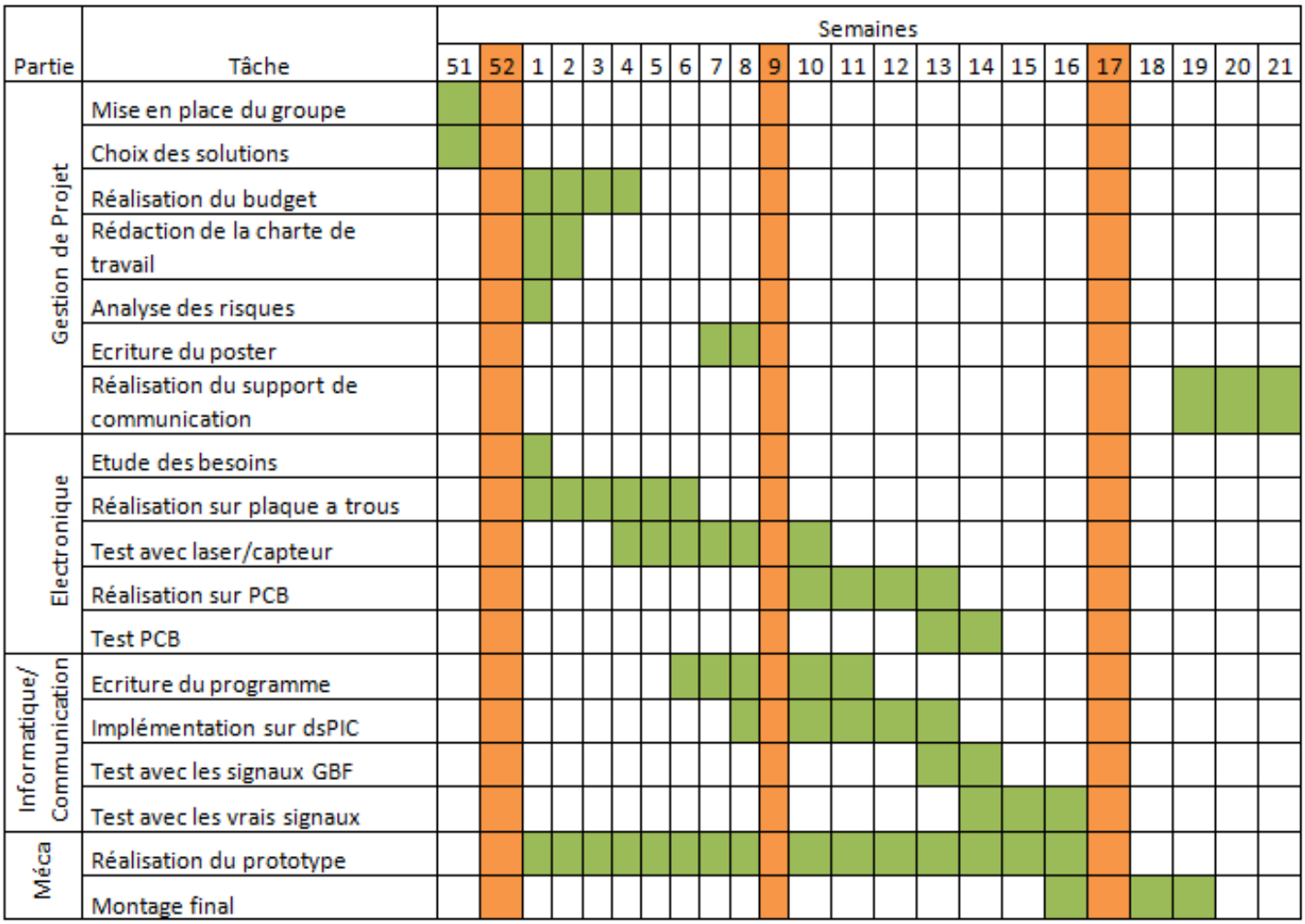


Photo légendée de la tourelle

Diagramme de GANTT :



Evaluation latérale :

Nom	Membre 1	Membre 2	Membre 3	Moyenne
Aziz	13.05	13	17.05	14.37
Eric	15.76	13.35	18.05	15.72
Sahbi	16.05	14.17	17.94	16.05
Adrien	16.23	16.82	15.17	16.07