

MINISTÈRE DES ÉTUDES SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE D'ALGER

DÉPARTEMENT ÉLECTRONIQUE



المدرسة الوطنية المتعددة التقنيات  
Ecole Nationale Polytechnique

---

Projet N ° 2 TTC

---

FERHAT Hiba El Batoul

Juillet 2021

---

# PARTIE THÉORIQUE

---

De nos jours, les combustibles fossiles couvrent la totalité d'énergie consommée. Or que ces derniers deviennent de plus en plus rares, pendant que les demandes énergétiques du monde s'élèvent exponentiellement. Il est estimé que les réserves mondiales seront épuisées d'ici quelques décennies si on ne change pas radicalement de ressource.

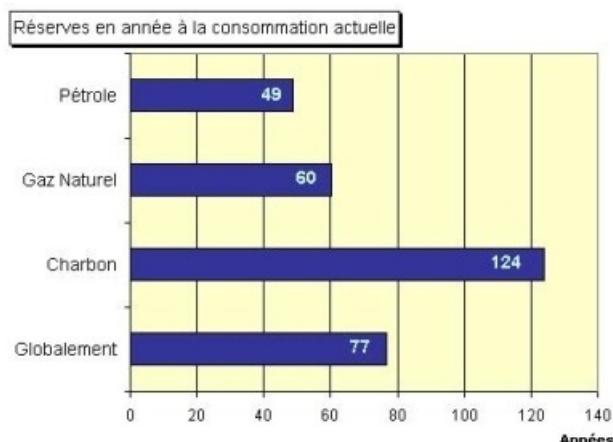


FIGURE 1: Les estimations des quantités de réserve restante

A notre niveau il est indispensable de trouver des alternatives. Les contraintes imposées et de trouver une source d'énergie économique, durable et peu polluante et étant donné que le sahara couvre 84% de la superficie de notre pays, le choix idéal qui répond aux contraintes est l'énergie solaire, une énergie abondante, inépuisable et facilement exploitable.

Pour pouvoir optimiser la collecte d'énergie solaire, faut bien comprendre le fonctionnement unitaire. Une cellule photovoltaïque est réalisée à partir de deux couches de silicium, une dopée P (dopée au bore) et l'autre dopée N (dopée au phosphore) créant ainsi une jonction PN avec une barrière de potentiel. Cette dernière est directement exposée sans couvert ce qui la rend photosensible, lorsque les photons sont absorbés par le semi-conducteur, ils transmettent leur

énergie aux atomes de la jonction PN de telle sorte que les électrons de ces atomes se libèrent et créent de plus en plus des électrons (charges N) et des trous (charges P), ceci crée encore une différence de potentiel plus importante entre les deux couches. Ce phénomène est appelé effet photovoltaïque.

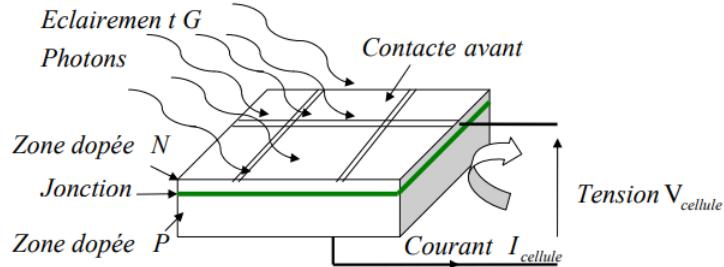


FIGURE 2: Principe de fonctionnement d'une cellule PV

Donc la puissance électrique produite par un panneau photovoltaïque dépend directement à l'intensité du flux lumineux. Mais cette dernière change selon la position du soleil qui varie au long de la journée. Afin d'optimiser la collection, on utilise des dispositifs de poursuite solaires ou « Trackers solaires ».

## Principe de fonctionnement

---

Un tracker est un système utilisant le principe de l'héliostat. C'est une structure portante qui permet d'orienter les capteurs solaires constamment vers le soleil en compensant le mouvement de rotation de la terre par un mécanisme d'horlogerie d'une manière perpendiculaire et en temps réel. Ceci permet d'augmenter considérablement l'énergie emmagasinée.

On distingue deux classifications :

### Classification selon le mécanisme de fonctionnement

---

#### *Système à un seul axe (mono-axe) :*

Le suiveur solaire tourne autour d'un seul axe. Le trajet suivi par cet axe est souvent en azimut qui est la rotation horizontale du soleil, c'est-à-dire d'est en ouest au fil de la journée. L'angle d'inclinaison du panneau est fixé selon la latitude du lieu où est disposé le capteur, ainsi elle reste constante au cours de son fonctionnement. Les systèmes à un seul axe sont les moins coûteux, leur commande est facile à mettre en œuvre, par contre leur efficacité est moyenne.

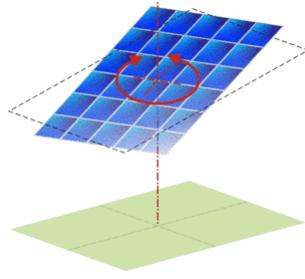


FIGURE 3: Un suiveur solaire mono-axe

#### *Système a double axes :*

Contrairement au précédent, ce suiveur solaire possède deux axes de rotation ce qui lui permet de suivre les mouvements en azimut (horizontalement) et en inclinaison (verticalement) au cours de la journée. Ainsi, le capteur reste constamment orienté vers le soleil ce qui garantie une meilleur production énergétique. Ces systèmes sont les plus performants mais les plus difficiles à commander.

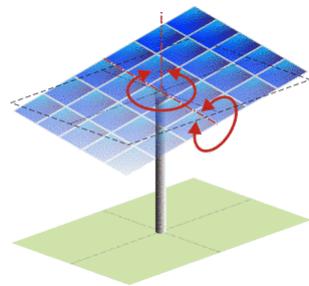


FIGURE 4: Un suiveur solaire bi-axe

---

#### *Classification par principe et méthodologie de fonctionnement*

Maintenant, on s'intéresse a la méthode détection du mouvement solaire, ceci se fait de deux manières :

#### *Méthode astronomique :*

Cette méthode préprogrammée qui repose sur le calcul de la position du soleil en se basant sur les équations géométriques et astronomiques. Le système nécessite comme entrées, lors de l'installation, la latitude et la longitude du site, la date du jour et l'heure. Cette méthode est très précise mais très complexe a réalisée donc elle est à éviter.

## *Méthode des capteurs de lumière :*

En utilisant des capteurs d'impédance sensible à la luminosité dont la variation change les proportions de la tension en sortie. Cette méthode est plus simple à réaliser et surtout permet une orientation optimale du panneau indépendamment de la zone dans laquelle le système sera situé.

## Analyse des schémas

---

### *Synoptique*

---

Mon choix se porte sur la réalisation d'un système à double axes avec la méthode des capteurs de lumière. Ainsi je résume le fonctionnement par :

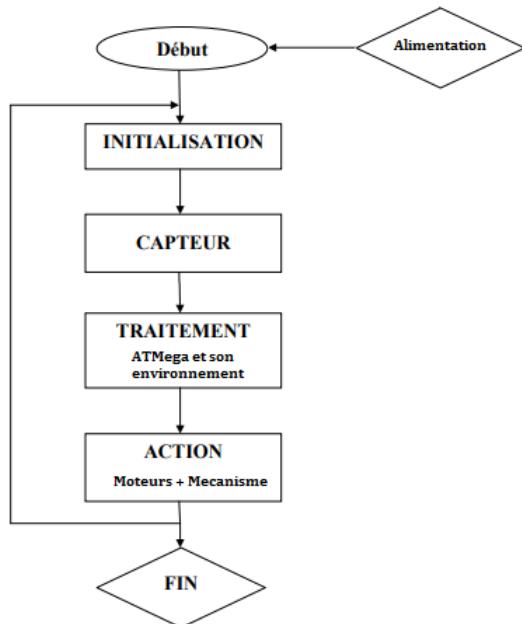


FIGURE 5: Algorithme de fonctionnement

Il admet la décomposition suivante :

### *ATMega328P :*

C'est le bloc de commande du système entier, c'est un choix peu populaire car il est généralement intégré dans les cartes Arduino qui est une plateforme prête à exploiter contrairement à l'ATMega qui nécessite un environnement d'adaptation.

- **Constitution et caractéristiques :** Voici à quoi ressemble un ATMega328P :

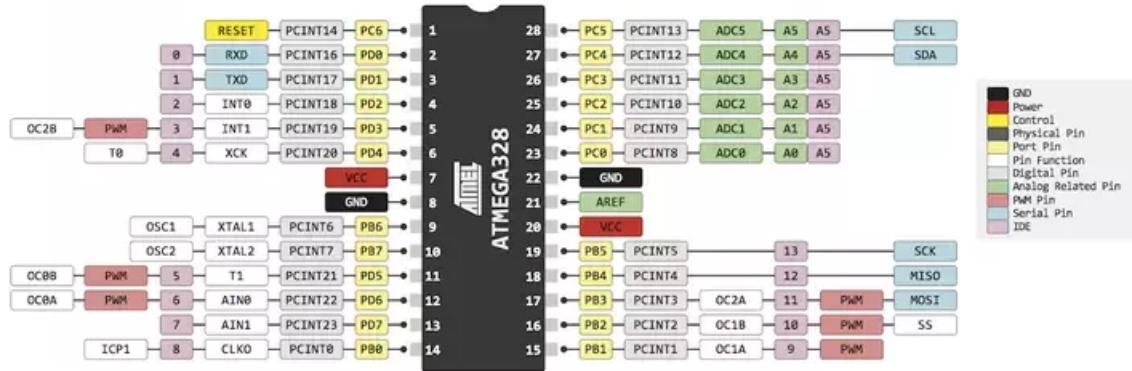


FIGURE 6: Description des entrées sortie du ATMega328P

- Canal de 8 bits.
- Une architecture RISC qui combine une mémoire Flash ISP de 32 Ko avec des capacités de lecture et d'écriture.
- Une EEPROM de 1024 Ko.
- Une SRAM de 2 Ko.
- 23 lignes d'E/S à usage général.
- 32 registres de travail à usage général.
- 3 compteurs/horloges flexibles avec des modes de comparaison.
- Des interruptions internes et externes.
- Un USART programmable en série.
- Un port série SPI.
- Un convertisseur A/N 10 bits à 6 canaux (8 canaux dans les boîtiers TQFP et QFN/MLF).
- Une minuterie chien de garde programmable avec oscillateur interne et 5 modes d'économie d'énergie sélectionnables par logiciel.
- Fonctionnement entre 1,8 et 5,5 V, selon le constructeur.
- Des débits proches de 1 MIPS par MHz en équilibrant la consommation d'énergie et la vitesse de traitement.
- **Pourquoi ce choix :** Ce n'est pas vraiment un choix car c'est une exigence imposée mais on peut la justifier par le fait qu'il s'agit d'un microprocesseur de bonne rapport qualité prix (bonne performance pour un prix relativement bas), une faible consommation d'énergie, la disponibilité de documentation, c'est module publié sous licence créative commons, il peut être exploité comme un Arduino et donc bénéficier également des avantages de ce dernier.

### *L'environnement d'adaptation du ATMega328P :*

Comme les ATMega sont largement utilisé avec les cartes de développement comme ESP32, Raspberry-pi mais surtout les Arduino comme ce processeur constitue l'élément principale de la carte. Il est donc possible de concevoir un environnement équivalent de ce dernier de sorte de pouvoir programmer notre microprocesseur en C/C++ (J'ai préféré le codage en C, car je dispose une grande maîtrise de ce dernier). La création de l'environnement passe par deux parties :

- **La partie software :** Elle prend en charge l'implantation du Bootloader qui est l'équivalent a l'ajout du système d'exploitation qui bien celui du Arduino, sans Bootloader aucun programme n'est fonctionnel. Une fois le Bootloader est supporté par le processeur on peut le programmé sur n'importe code du système et qu'il reste figée jusqu'à la prochaine programmation.
- **La partie hardware :** Elle consiste a concevoir le circuit qui permet d'avoir le mémé fonctionnement d'un Arduino.

### *Le capteur :*

C'est l'élément qui est sensé être sensible aux variations de luminosité et dans quel sens ce fait cette variation, en les traduisant en tension de sortie. Ça peut être a base des phototransistors, des opto-coupleurs...

### *Les moteurs :*

On aura besoin de deux moteurs pour les deux axes, le mouvement horizontal (en azimut) et le mouvement vertical (en altitude). Les choix possibles sont très nombreux.

### *La partie mécanique*

C'est la partie la plus consistante, malheureusement aucune conception sur internet (ou du moins celles que j'ai rencontré) ne sont bien faite, elles sont majoritairement mal dimensionnées ou mal conçues et qui dit un changement de mécanisme dit un changement du code! Honnêtement je trouve que c'est la partie la plus difficile. J'ai due recorrigé l'un des systèmes trouvé sur internet et ceci ma demander beaucoup de temps et de tentatives a cause de manque de maîtrise de Solidworks.

## *L'alimentation :*

Le choix de l'alimentation est très important non seulement pour le fonctionnement mais pour la sécurité du circuit, un mauvais choix risque de griller le travail entier. On cherche à trouver une source de tension constante et stable idéalement de 5V comme le constructeur indique.

## *Schémas partiels*

Traitant les cas possibles pour chaque bloc avant de trouver la combinaison idéale :

### *ATMega328P et son environnement :*

L'exploitation directe du microprocesseur vierge n'étant pas possible il faut d'abord passer par deux adaptations déjà précisées :

- **Adaptation software :** Celle-ci a son tour est répartie en deux :

- Chargement du Bootloader : qui est l'équivalent de l'implantation du système d'exploitation, ce logiciel permet le lancement ainsi que le chargement d'un sketch dans la mémoire FLASH du microcontrôleur à partir de la ligne série. Pour ce faire il existe de nombreuses méthodes par deux Arduino, par un Arduino et un environnement physique ou par une clé USB dit USBASP.

Les deux premières méthodes sont équivalentes (car l'environnement physique donne un fonctionnement que l'architecture d'un Arduino) en utilisant le mode ArduinoISP<sup>1</sup> et avec le montage suivant charge le programme ISP :

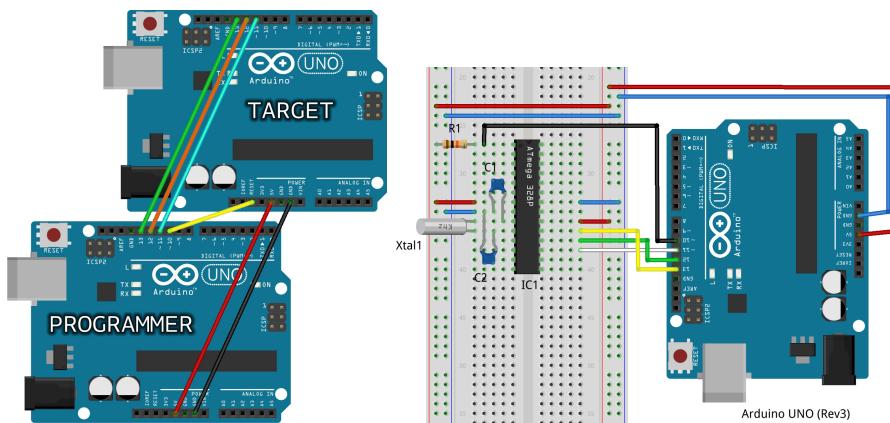


FIGURE 7: Deux montages de Bootloader à deux et à un seul Arduino

1. Abréviation de "In System Programming", il est dit également "In Circuit Serial Programming". Il est très rare de programmer des circuits intégrés avant qu'ils ne soient soudés sur un PCB. Au lieu de cela, la plupart des microcontrôleurs ont ce qu'on appelle un en-tête de programmation in-system (ISP). En particulier, certains fabricants de circuits intégrés, comme Atmel et Microchip, ont une méthode ISP spécialisée pour programmer leurs circuits intégrés. C'est ce qu'on appelle la programmation série en circuit (ICSP). La plupart des cartes Arduino et compatibles Arduino ont un connecteur ICSP à 2x3 broches. Ce sont les broches auxquelles vous devrez connecter votre programmeur afin de reflasher le firmware de votre carte.

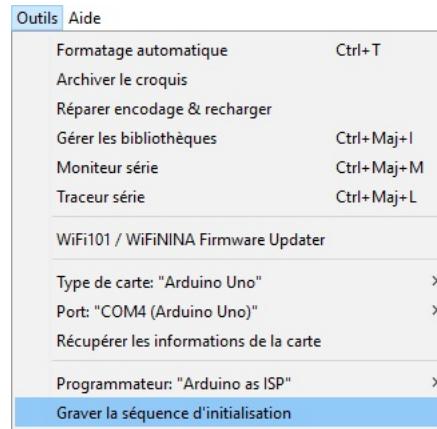


FIGURE 8: Les configurations a mettre avant de graver

La méthode de USBASP est la plus simple car elle nécessite pas de montage mais des drivers et des bibliographie seulement puis un téléversement direct :

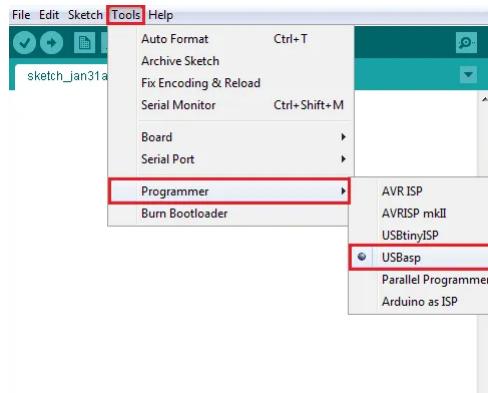
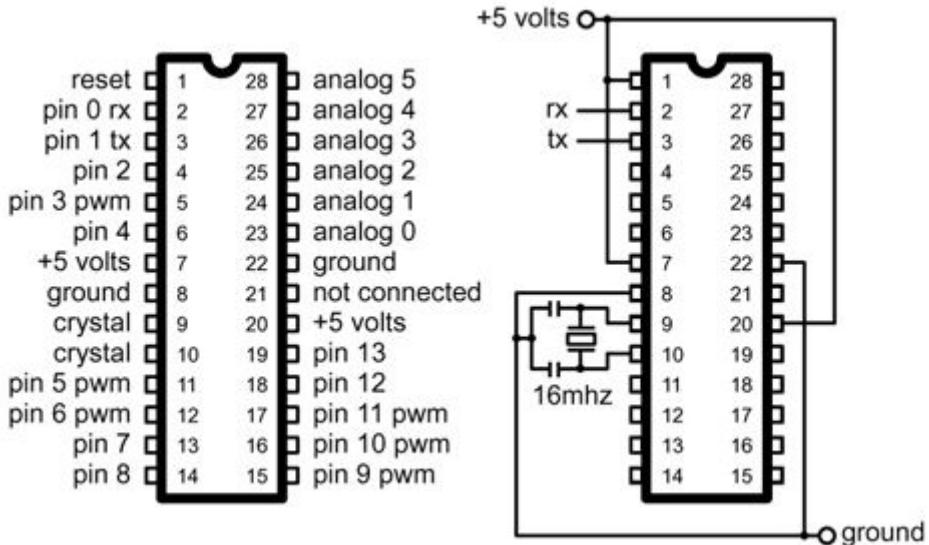


FIGURE 9: Les configurations a mettre avant de graver

- Téléversement du programme : Une fois le chip est bootloader il ne reste plus qu'à implanter le programme avec qui elle reste figée jusqu'à la prochaine programmation. On peut utiliser le support de la carte Arduino avec notre chip et on téléverse notre sketch.
- **Adaptation hardware :** C'est le montage permettant d'émulé le fonctionnement du Arduino Uno afin de créer un mappage entre les pins de du ATMega et ceux du arduino uno. Il existe plusieurs configurations selon les propriétés souhaitées, j'ai choisi le montage suivant :



Les principaux éléments ajoutés sont bien le quartz  $16GHz$  et les deux capacités  $22pF$  pour créer un oscillateur externe permettant d'avoir un meilleur fonctionnement. Ces éléments sont élémentaires et on peut se contenter de l'oscillateur interne du ATM qui vaut  $8GHz$ . Il faut noté aussi la possibilité d'ajout d'une capacité de  $10nF$  entre la source et la masse pour réduire les effets parasites.

### *Le capteur :*

Les choix des composantes du capteur sont multiples, on distingue principalement :

- **Les phototransistors :** Elles ne sont pas très appréciées pas que pour leurs mauvais rendement mais aussi leurs sensibilités aux grandeurs d'influences particulièrement la température.
- **Les photodiodes et les opto-coupleurs :** Ils ont le même effet (circuit fermé en présence de lumière et circuit fermé en son absence) mais ils sont plus performants en matière de rapidité, résilience et rendement.
- **Les LDRs :** Une photo résistance LDR (Light Dépendent Résistor ou résistance dépendant de la lumière) est un composant dont la résistivité est sensible à la lumière à laquelle il est exposé (très grande résistivité en obscurité et une faible résistivité en luminosité). Elle est fortement concurrencée par la photodiode. J'ai choisi la LDR pour son temps de réponse réduit. La rapidité de réaction est indispensable pour ce projet.

Le montage permettant d'exploiter correctement les LDR est un pont diviseur de tension mais il faut que les LDRs soient disposées de manière à avoir qu'une seul (une idéalement ou deux au maximum) LDR touché par la lumière à un temps donné car ces derniers indique le sens d'orientation du système pour y faire on utilise une séparation de sorte que si la lumière touche une LDR les autres sont couvertes par l'ombre. Ainsi, les proportion de tension créée au niveau de la LDR touché permet au microprocesseur d'identifier l'orientation, le système atteint son

état stable qu'une fois toutes les LDRs sont exposées équitablement a la lumière ce cas convient bien la perpendicularité au flux lumineux qui optimise la collecte d'énergie. Pour visualiser cette séparation passer a la partie mécanique.

### *Les moteurs :*

Comme il s'agit d'un système simple de prototypage en plastique il préférable d'utiliser un moteur léger et économique, un servo-moteur suffit largement surtout qu'il a un bon angle d'écart de  $180^\circ$  et un couple important par rapport a nos besoins.

### *Partie mécanique :*

J'ai finit par choisir ce mécanisme :



FIGURE 10: Conception mécanique du système

Comme vous voyer une bonne partie du circuit est mobile et non pas fixe. Voila mon résultat :



### ***L'alimentation :***

Comme c'est mentionné dans le datasheet le système nécessite une tension alternative de 5V idéalement, une batterie ne convient pas le montage encore moins le secteur. J'ai pensé à utiliser d'un transformateur qui n'est rien d'autre qu'un simple chargeur avec un port adapté au circuit.

### ***Schémas globale***

---

Après avoir discuter les cas possibles, voilà ce que j'ai aboutie au final :

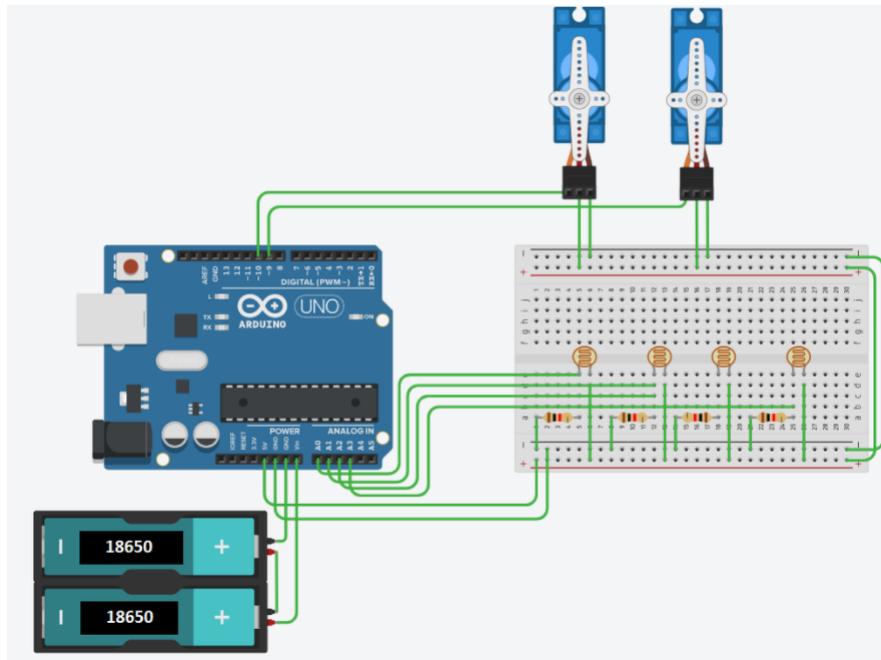
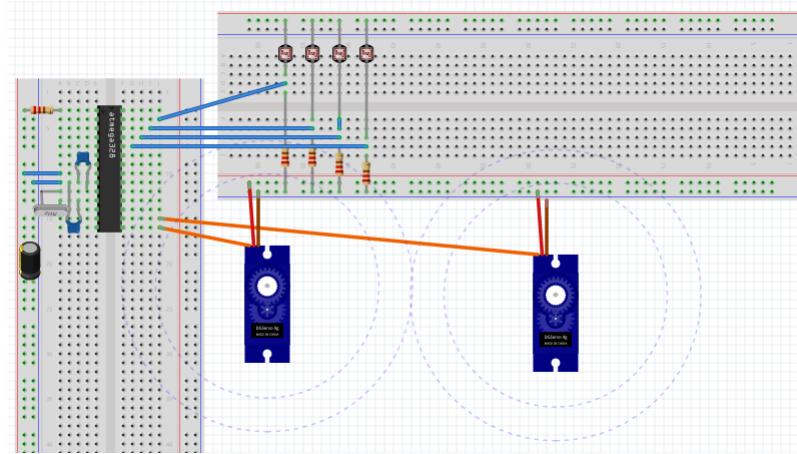


FIGURE 11: Montage final Post-Analyse

Il faut noté que j'ai mis un Arduino juste a titre d'illustration car je ne trouver pas l'ATMéga328P dans fritzing mais seulement l'ATMéga328, donc je réalise le monatage suivant seulement a titre indicatif.



## Programmation du micro-processeur ATMega328P

---

Après avoir bien préparer l'environnement et réaliser le montage, j'ai conçue deux codes différents (pour les deux montages précédents) car je voulais comparer entre les deux codes :

### *Premier code*

---

C'est le code du montage qui permet le réglage potentiométrique des vitesses des servo-moteurs.

```

1 #include <Servo.h>

3 Servo horizontal; // Le servo horizontal
4 int servoh = 180;
5 int servohLimitHigh = 175;
6 int servohLimitLow = 5;
7 // 65 degrees MAX

9 Servo vertical; // Le servo vertical
10 int servov = 45;
11 int servovLimitHigh = 50;
12 int servovLimitLow = 5;
13

14 // Connection des pins
15 // nom de la pin = analog pin;
16 int ldrlt = A0; //LDR haut gauche
17 int ldrrt = A2; //LDR haut droit
18 int ldrld = A1; //LDR bas gauche
19 int ldrrd = A3; //LDR bas droit

21 void setup(){
22     horizontal.attach(9); // Pin relie au servo horizontal
23     horizontal.write(180); // Angle maximale de rotation du moteur horizontal

25     vertical.attach(10); // Pin relie au servo vertical
26     vertical.write(45); // Angle maximale de rotation du moteur vertical
27     delay(2500);
28 }

29

30 void loop() {
31     int lt = analogRead(ldrlt); // La proportion de tension cause par les variation
32         de la resistance de la LDR haut gauche
33     int rt = analogRead(ldrrt); // La proportion de tension cause par les variation
34         de la resistance de la LDR haut droit
35     int ld = analogRead(ldrld); // La proportion de tension cause par les variation
36         de la resistance de la LDR bas gauche
37     int rd = analogRead(ldrrd); // La proportion de tension cause par les variation
38         de la resistance de la LDR bas droit

39     int dtime = 10; // Resolution temporelle
40     int tol = 50; // Tolerance pour laquelle les moteurs changent de rotation

41     int avt = (lt + rt) / 2; // La moyenne de la proportion de tension cause par les
42         LDR en haut
43     int avd = (ld + rd) / 2; // La moyenne de la proportion de tension cause par les
44         LDR en bas
45     int avl = (lt + ld) / 2; // La moyenne de la proportion de tension cause par les
46         LDR de gauche

```

```

int avr = (rt + rd) / 2; // La moyenne de la proportion de tension cause par les
    LDR de droite
43
int dvert = avt - avd; // La difference de moyenne entre le haut et le bas, si c
    'est positif la rotation en haut sinon la rotation en bas.
45 int dhoriz = avl - avr; // La difference de moyenne entre le droit et le gauche,
    si c'est positif la rotation a gauche sinon la rotation a droit.

47 if (-1*tol > dvert || dvert > tol) // Verification la tolerence angulaire dans
    le mouvement vertical, si elle est atteinte on change d'angle
{
49    if (avt > avd) // Il y a plus de luminosit en haut qu'en bas
    { servov = ++servov; // Servo doit avoir une rotation negative
51        if (servov > servovLimitHigh)
    { servov = servovLimitHigh;} // Limite la rotation du servo pour des raisons
        de securite
53    }
    else if (avt < avd) // Il y a plus de luminosit en bas qu'en haut
55    { servov= --servov; // Servo doit avoir une rotation negative
        if (servov < servovLimitLow)
    { servov = servovLimitLow;} // Limite la rotation du servo pour des raisons
        de securite
57    }
59    vertical.write(servov);}

61 if (-1*tol > dhoriz || dhoriz > tol) // Verification la tolerence angulaire dans
    le mouvement vertical, si elle est atteinte on change d'angle
{
63    if (avl > avr) // Il y a plus de luminosit a gauche qu'a droite
    { servoh = --servoh; // Servo doit avoir une rotation negative
65        if (servoh < servohLimitLow)
    { servoh = servohLimitLow; } // Limite la rotation du servo pour des raisons
        de securite
67    }
    else if (avl < avr) // Il y a plus de luminosit a droite qu'a gauche
69    { servoh = ++servoh; // Servo doit avoir une rotation positif
        if (servoh > servohLimitHigh)
    { servoh = servohLimitHigh; } // Limite la rotation du servo pour des raisons
        de securite
71    }
73    else if (avl = avr)
    { delay(5000); } // Si la luminosit de part et d'autre est la meme donc le
        tracker est bien centralis , un delais de 5s avant de reverifier
75    horizontal.write(servoh);
77    }

    delay(dtime);}
```

## Second code

---

C'est le code du montage sans réglage de vitesse, la vitesse est par conséquent constante.

```
#include<Servo.h>
2
3 Servo alt, az; // Defenir les variable des tensions controlant le servo.
4 int alt_Deg, az_Deg; // Les angles des servo en degré.
5 const byte left=A0, right=A1, top=A2, bottom=A3; // Introduire les pins des
6           diviseurs de tension controle par les LDRs.
7 int L_value, R_value, T_value, B_value; // La valeur des diviseurs de tension
8           caus par les LDRs.

9 int x,y;

10 void setup()
11 {
12     Serial.begin(9600);
13     az.attach(9); // C'est le pin du servo controlant le mouvement horizontal.
14     alt.attach(10); // C'est le pin du servo controlant le mouvement vertical.

15     // initialisation du servo to a une position initiale (90,30).
16     Serial.println("SETTING UP SERVO");
17     alt_Deg=20; az_Deg=20;
18     az.write(90);
19     alt.write(30);
20     delay(1000);
21 }

22 }

23 void loop() {
24     // Impose les variation maximale des servo par mesure de sécurité.
25     if(az_Deg<=5)
26         az_Deg=5;
27     if(az_Deg>=175)
28         az_Deg=175;
29     if(alt_Deg<=5)
30         alt_Deg=30;
31     if(alt_Deg>=150)
32         alt_Deg=150;

33
34     // Lecture des valeurs de tension des pont des LDRs.
35     L_value=analogRead(left);
36     R_value=analogRead(right);
37     T_value=analogRead(top);
38     B_value=analogRead(bottom);

39
40     // Identifier les diff'rence de tension de sorte de trouve la luminaunit la
41           plus importante.
42     x=L_value-R_value;
```

```

y=B_value-T_value;

44
// La variation de position si la r solution est de 10.

46 if(x<-7)
    az_Deg+=1;
48 if(x>7)
    az_Deg-=1;
50 if(y<-7)
    alt_Deg+=1;
52 if(y>7)
    alt_Deg-=1;
54 az.write(az_Deg);
    alt.write(alt_Deg);
56 delay(70); // S'il retrouve bien la position optimale sinon il reprend l'
               algorithme.
}

```

2.ino

Après les avoir tester chez moi, les deux montages sont pratiquement équivalent et la variation de vitesse est pratiquement insensée. Pour cela je préfère évidemment le second cas nécessite moins de composants.

---

# PARTIE PRATIQUE

---

## Simulation

---

Comme mon circuit comprend un micropocessuer qui est malheureusement non supporté par les simulateur conventionnel comme multisim et orcad ni même pas fritzing, je suis contrainte de trouver un simulateur la plupart des propositions sur internet son autour d'un logiciel dit Atmel mais ce dernier est payant et j'ai pas trouver un bon craque. Donc la simulation n'est pas possible.

## Réalisation du PCB

---

### *Étapes de réalisation*

---

La réalisation du circuit imprimé doit respecter certaines règles :

- Respecter le format standard des cartes.
- Placement des composants impérativement sur la grille standard (pas de 2.54mm).
- Alignement des composants.
- Mettre a part les composant lourds sans les placés sur le circuit imprimé.
- Miniaturisation de la surface du circuit.
- De préférence essayée de faire un simple face.

### *Le circuit imprimé du montage*

---

J'ai utilisé Multisim pour la création du PCB, après avoir créer un fichier de simulation :

- On commence avec un bon repérage.
- On doit affecté à chaque composant un footprint (empreint) qui correspond au composant physique que nous disposant. J'ai utilisé un autre fichier différent que celui de la simulation dont car j'ai utilisé les composant physique, ce processus est lourd car on ne peut pas voir graphiquement la forme du composant ni ses dimensions.
- Une fois la fenêtre apparaît, on visualise que les composants sont dispersés d'une manière aléatoire et on doit les réarrangé suivants les règles de notre cours. Pour nous faciliter la tache on peut commencer par un arrangement automatique proposé par le logiciel par l'opération suivante " Autoroute > Autoplace parts ".
- Choisir l'option " Silkscreen Bottom " et décocher la case de routage dans les propriétés.
- Maintenant on doit les réarrangé correctement.
- Maintenant le routage par " Autoroute > Start/resume autorouter ".

Le PCB du montage retenu :

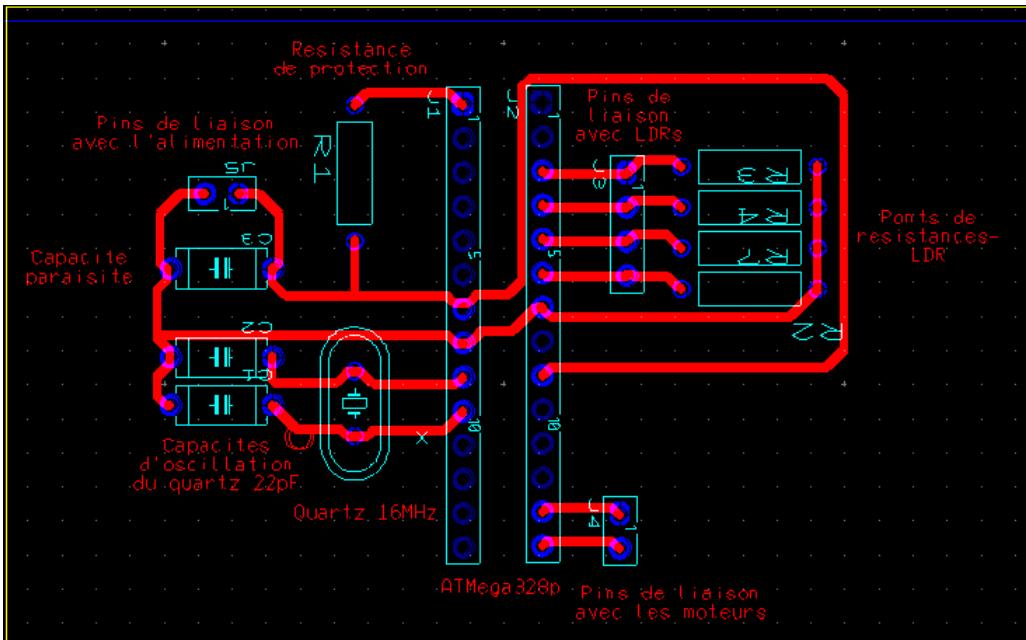


FIGURE 12: Le circuit imprimé du montage

#### Remarque :

- A propos le choix des composants, comme le DIP28 de multisim a une largeur qui ne correspond pas celle du ATMega ou son support j'ai utiliser des pins. Le fait qu'ils soient mal ou femelle n'a pas d'impact comme les dimension sont les mêmes.
- Certain composants ne sont pas au niveau du PCB, comme le pont des LDRs car ces derniers ne sont pas fixes mais en mouvement donc j'ai ajouté des pins pour les insérer.

#### 0.0.1 ) Le model 3D

---

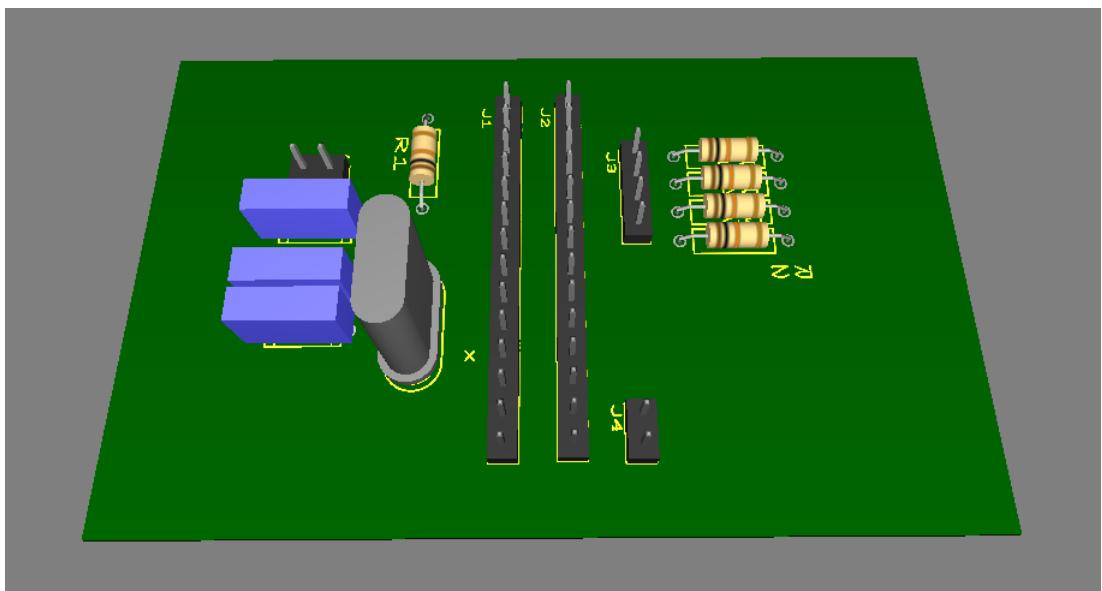
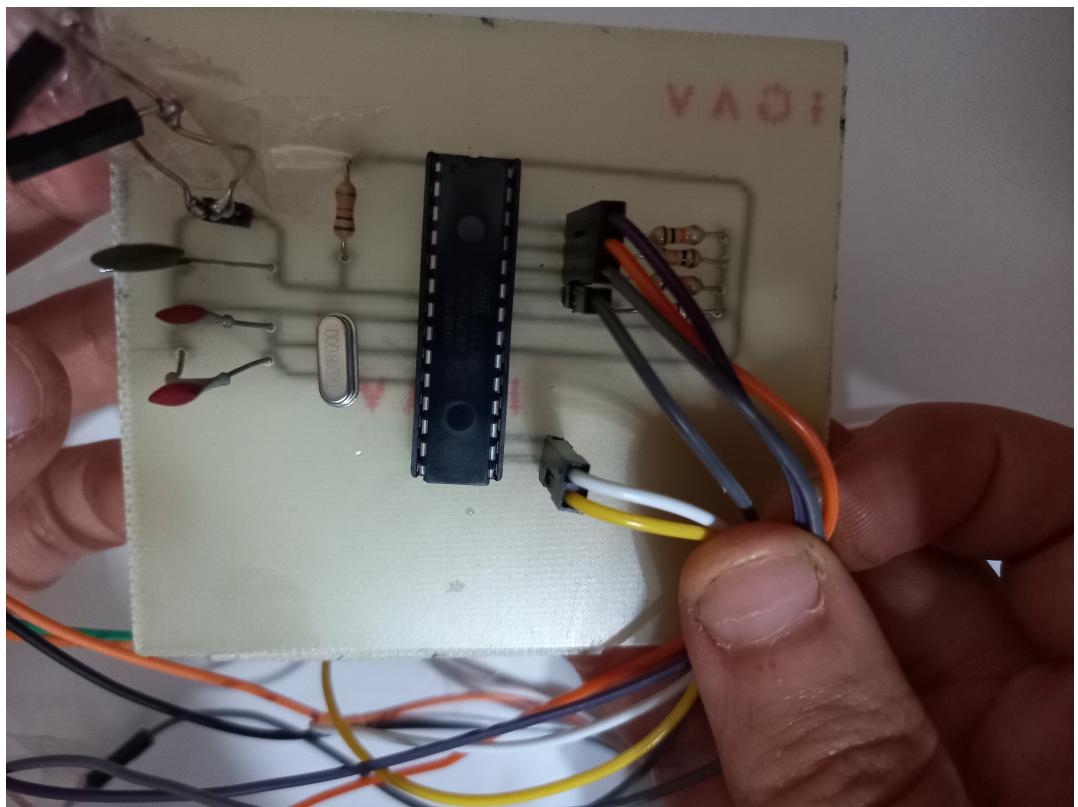
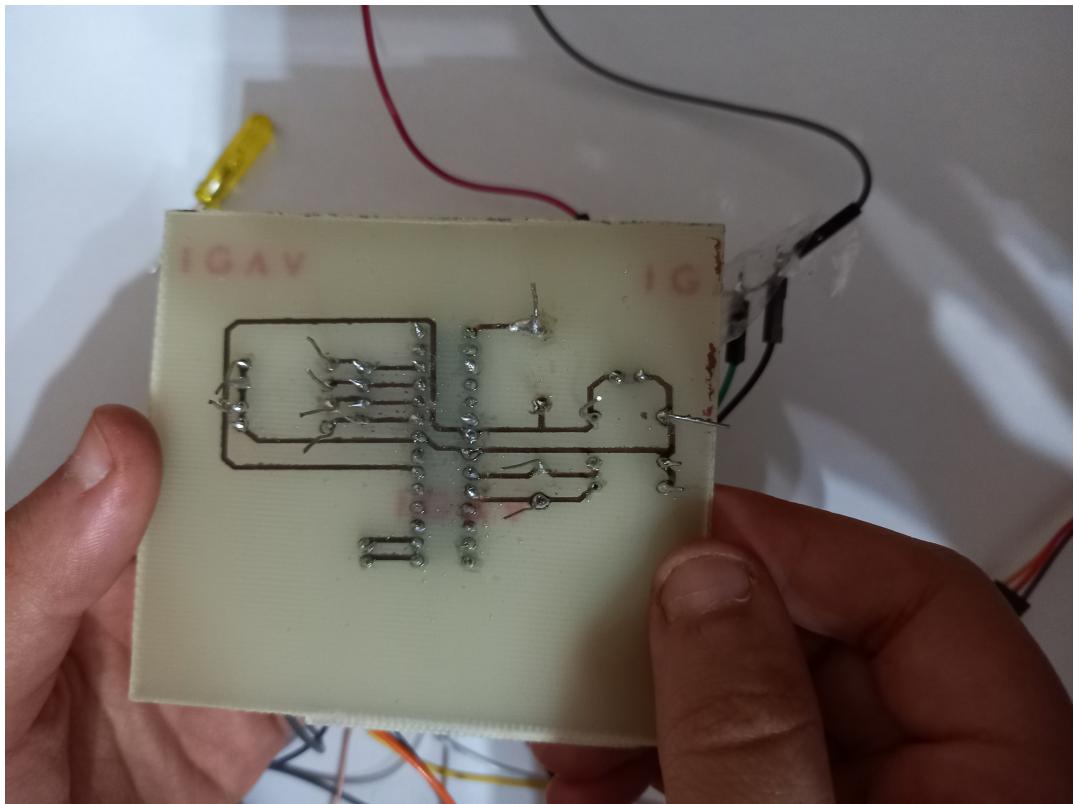


FIGURE 13: Le modele 3D du PCB

### *PCB réalisé*

---

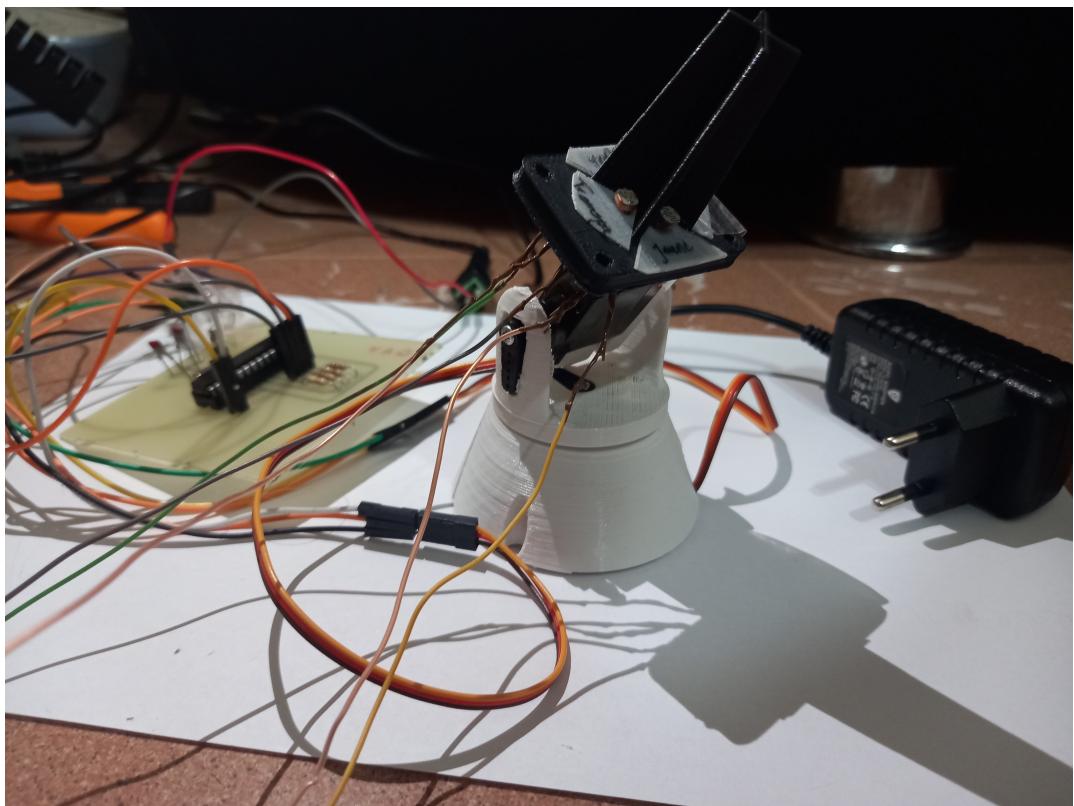




## Le prototype final

---

La vidéo montrant son fonctionnement est envoyée en pièce jointe de l'email.



---

## CONCLUSION GÉNÉRAL

---

Grâce a ce projet j'ai développer de nombreuses compétences :

- La bonne méthode de réalisation des PCB voire la maîtrise du processus (après plus de 7 tentatives non réussites).
- Reprendre mes compétences perdues en matière CAO et conception 3D ou j'ai reconcu mes propres pièces avec les dimensions qui me conviennes.
- Réalisation d'un prototype " From Scratch " et ainsi comment faire une bonne analyse pour aboutir a un prototype fonctionnel.
- Comment passer d'une carte de développement a un microprocesseur propre, ceci est extrêmement important surtout si le prototype est destinée a la production industrielle en masse.
- Gestion de temps et la gestion du stresse surtout.