université de BORDEAUX

Rapport de stage

Configuration d'un héliostat

Cohabit Fablab

Thomas Pallaro

Professeur réferent : Stephane Ygorra

Tuteur de stage : Jean-Baptiste Bonnemaison

2019

Remerciement

Dans un premier temps, j'aimerais remercier Monsieur François Augereau, enseignant à l'IUT Génie Electrique et Informatique Industrielle de m'avoir offert l'opportunité d'effectuer mon stage au Fablab. C'est lui qui m'a présenté à l'équipe de l'association et proposé le sujet sur lequel j'ai travaillé.

Je voudrais aussi remercier Jean-Baptiste Bonnemaison et Pierre Grange Praderas de m'avoir accepté et de m'avoir supervisé au cours de mon projet, ainsi que Thierry qui m'a supervisé également. Je remercie aussi toutes les personnes en service civique qui étaient présentes au Fablab durant mon stage et qui m'ont intégrées au sein de l'équipe et aidé lors de l'utilisation de machines comme l'imprimante 3D.

Résumé

Un héliostat ou traqueur solaire est un dispositif permettant de suivre la course du Soleil, généralement pour orienter toute la journée les rayons solaires vers un point ou une petite surface fixe à l'aide de miroirs. Cela permet d'éclairer une pièce qui serait isolée du soleil ou de chauffer un objet, par exemple un four solaire. On peut également y mettre des panneaux solaires pour pouvoir produire de l'électricité tout au long de la journée.

An heliostat or solar tracker is a device who allow to follow the sun trajectory, usually to orient all the day the suns rays to a fix point, with mirror. That's allow to shine an isolate room or heat an object, for example a solar furnace. We can also put solar pannels on it in order to produce electricity all the day.

Sommaire

Remerciement 2
Résumé 3
Introduction 5
Cohabit Fablab 6
Le projet 7
Reprise du projet 9
Circuit électrique 10
Apport au projet 11
Création de la pièce 3D 11
Création d'une carte 11
Programmation de la carte 14
Objectif d'amélioration 15
Difficultés rencontrées 16
Conclusion 17
Index 18
Annexe 1 19

Introduction

Du 6 Mai au 6 Juin, j'ai réalisé un stage au sein d'une association de l'université de Bordeaux, le Cohabit Fablab. Durant ce stage j'ai pu mettre en pratique mon savoir théorique au sein d'un projet.

Ce stage était pour moi une chance de pouvoir réaliser un projet de manière autonome, de découvrir de nouvelles choses et de travailler au sein d'une association avec des personnes expérimentées. Le fait de pouvoir créer ma propre carte et de la programmer m'a énormément attiré et c'est une des raisons pour lesquelles j'ai demandé à effectuer mon stage à cet endroit. C'était aussi une occasion de pouvoir développer des compétences pratiques et d'apprendre à utiliser de nouveaux logiciels, ou de performer mes connaissances dans d'autres. J'ai aussi été en contact avec des personnes extérieures à l'université et pu discuter avec eux.

A part apprendre et découvrir de nouvelles choses, ce stage m'a permis de confirmer que j'aimais bien travailler dans le domaine de l'électronique et de l'informatique et à conforter mes choix d'orientations.

Mon stage au Fablab à consisté essentiellement à créer une carte et la programmer afin que l'héliostat qui était en ma possession puisse suivre la trajectoire du soleil.

Mon tuteur étant l'un des responsables de l'association, j'ai pu compter sur lui pour imprimer des pièces en 3D ou utiliser la découpe laser, j'ai pu également compter sur toutes les autres personnes présentes pour m'aider à m'imprégner du projet et dans la partie programmation.

Dans un premier temps je vais vous présenter l'association Cohabit Fablab, ensuite je vais vous présenter mon projet, comment je l'ai repris, ce que j'ai apporté et ce que je projetais de faire. Et je terminerai avec les difficultés que j'ai rencontrées.

Cohabit Fablab

Cohabit est une association appartenant à l'université de Bordeaux située 15 rue de Naudet à Gradignan, elle regroupe le Technoshop et le Fablab. Cette association permet d'accompagner des créateurs ou des entreprises lors d'un projet technique. Le Technoshop est la partie de l'association qui travaille avec les entreprises, quand au Fablab il est plus la pour travailler avec des particuliers ou des étudiants.

Certains IUT, comme l'IUT Génie Civile ou Génie Mécanique et Productique n'hésite pas à y envoyer leurs étudiants pour pouvoir compléter des projets. En effet le Fablab possède une découpe laser et plusieurs imprimantes 3D.

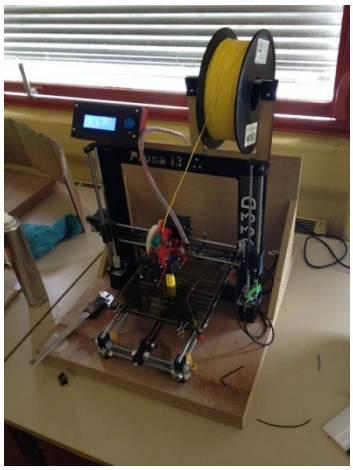

Figure 1 - Photo d'une imprimante 3D

Aujourd'hui le Président de l'association est Pierre Maunoury, il y a aussi deux Fab-managers qui sont présents à plein temps, Jean-Baptiste Bonnemaison et Pierre Praderas. Ce sont eux qui organisent les formations sur les machines comme la découpe laser, qui reçoivent les particuliers ou les étudiants pour s'associer à un projet, qui encadrent les stagiaires ou les personnes qui font un service civique.

Lorsque j'y étais, il y avait 4 personnes en service civique, Alexandre, Marvin, Sara et Florent. Ce sont eux qui s'occupent de faire vivre le Fablab, qui reçoivent les personnes qui veulent utiliser les imprimantes 3D, ou qui s'occupent de l'association si jamais les deux Fab-managers ne sont pas présents.

Le projet

Mon travail durant ce stage a consisté à reprendre et poursuivre le projet d'une étudiante qui était la avant moi. Ce projet c'est celui de l'héliostat, comme dit plus haut, un héliostat est un dispositif permettant de suivre la course du Soleil. L'objectif du projet consistait en une conception et réalisation d'un héliostat (suiveur solaire ou renvoi solaire) réalisable avec les moyens d'un Fablab au coût minimum. A terme, ce projet vise à s'implémenter dans certains pays pour accompagner les premiers travaux dans des Fablabs à l'étranger. En effet, car un Fablab va ouvrir au Mali et le Cohabit de Bordeaux va lui envoyer plusieurs projets réalisés afin de l'aider à se développer.

Le cahier des charges était :

- L'héliostat doit pouvoir être réalisé avec des moyens simples
- Le coût doit être le plus bas possible
- L'électronique embarquée doit être simple et peu coûteuse

Une analyse fonctionnelle a été réalisé pour pouvoir se rendre compte de à qui le projet va profiter et de voir quelle fonction il va réaliser.

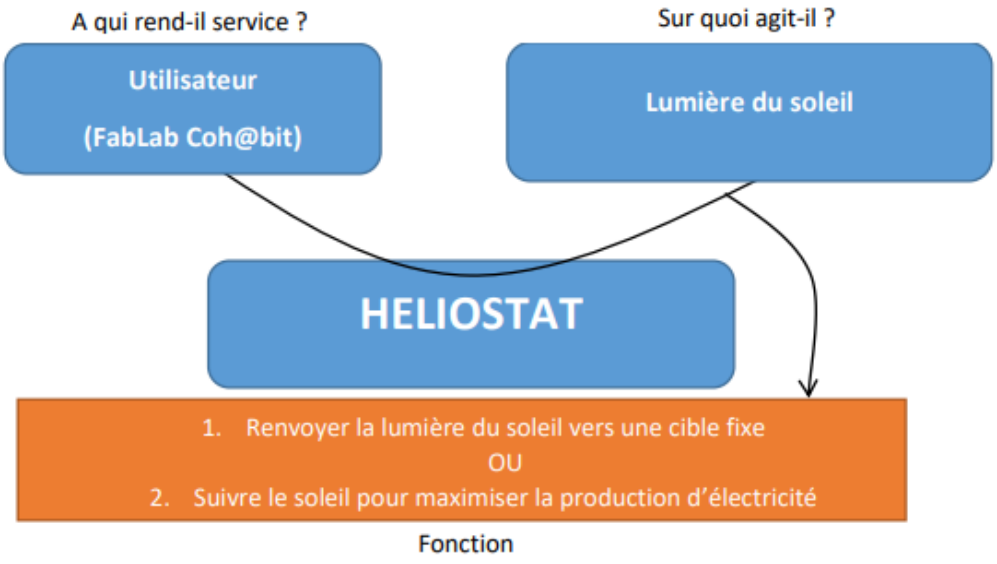

Figure 2 - Recherche du besoin

Figure 3 - Recherche des fonctions

Fonction	Description	Critères	Niveau	Flexibilité
FP1	Suivre le soleil	Angles d'azimut et d'élévation	+/- 1°	0
FC1	Alimentation	Type Source	Electrique Autoalimenté	0 0
FC2	Production électrique	Puissance Rendement	xx kWc (=prod. Max) XX	1
FC3	Conditions météorologiques	Pluie Vent	Etanchéité électrique Limiter la prise au vent	0 2
FC4	S'adapter à tout utilisateur	Montage Utilisation	Simple Simple	1 1
FC5	Fabrication	Machines / Outils Matériaux Coût	Disponible au FabLab Disponible au FabLab Minimal	1 1 1
FC5	Assurer la sécurité du système	Limites physiques ou programmées	Angles max/min	2
FC6	Assurer la sécurité de l'utilisateur	Arrêt d'urgence Protection électrique	Bouton d'arrêt d'urgence Isoler les parties électriques	2 1
FC7	Plaire à l'utilisateur	Forme Personnalisation	« Sympathique » Logo / Texte	3 3

FC8	Dimensions	Encombrement max Hauteur	30x30x30 cm Minimiser	2 2
FC9	Maintien au sol	Inclinaison du sol Type de sol	Plat Terre / Matériau dur	1 1
FC10	Maintenance système	Composants élec. Assemblage	Standards Démontable/remontable	1 1
FC11	Chaleur	Température de fonctionnement	Min : -10°C et - Max : 40°C et +	1 1

Reprise du projet

Lorsque j'ai repris le projet, la stagiaire avant moi avait déjà construit l'héliostat, fait un état de l'art du projet, commencé à programmer le microcontrôleur et fait quelques tests. Tout d'abord j'ai commencé par lire les documents qu'elle a laissés, les liens internet ou elle s'est inspirée pour construire l'appareil et le programmer et ensuite j'ai essayé de comprendre le programme arduino qu'elle avait fait et qui n'était pas tout à fait opérationnel.

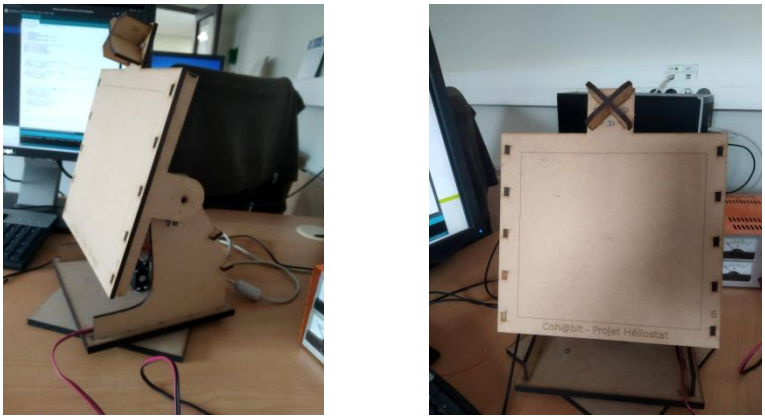

Figure 4 - l'héliostat

Lorsque j'ai fait les premiers tests, l'héliostat suivait bien la lumière de mon téléphone mais avec un décalage de 2 secondes environ. J'utilisais la lumière de mon téléphone pour simuler la trajectoire du soleil. La stabilisation n'était pas parfaite, l'héliostat continuait de bouger un peu, même quand la lumière était en face.

Circuit électrique

Le circuit est composé de deux servomoteurs, un microcontrôleur arduino, quatre Leds et huit résistances. Les Leds sont alimentées par la carte, les servomoteurs sont alimentés par une alimentation extérieure de 5V. Comme on peut le voir sur la figure si dessous (figure 7) les servomoteurs sont reliés à la carte sur les broches PWM, afin que le microcontrôleur puisse leur donner l'ordre de tourner. Les capteurs sont reliés sur les broches analogiques de la carte, car les données perçues par les capteurs sont analogiques et sont ensuite numérisées par le convertisseur qu'il y a sur la carte.

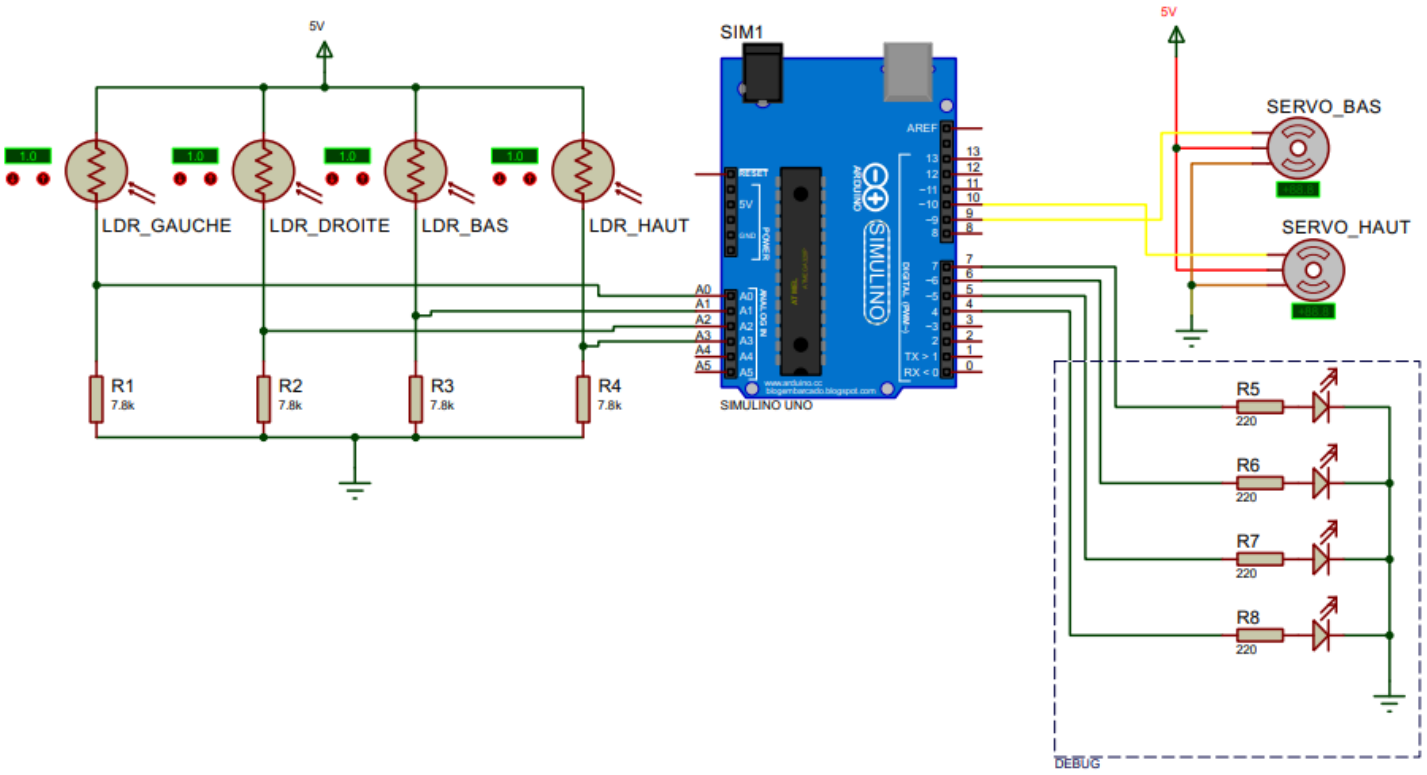

Figure 5 - Schéma électrique du montage

Apport au projet

Création de la pièce 3D

En premier, j'ai réalisé une pièce en 3D avec le logiciel Freecad. Cette pièce permet de mieux isoler les capteurs. En effet les capteurs étaient déjà séparés par une croix, mais cela n'est pas suffisant car le soleil peut éclairer les 4 capteurs avec la même intensité lumineuse même s'il n'est pas pile au dessus de l'appareil. Ce qui l'empêchera de fonctionner parfaitement.

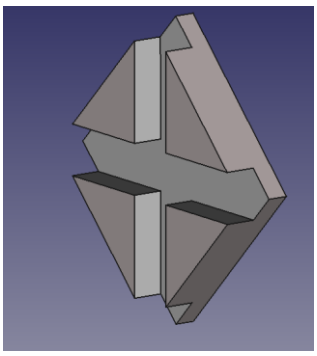

Figure 6 - Pièce sous Freecad

Figure 7 - Pièce sur l'héliostat

Du coup avec cette pièce les 4 capteurs seront éclairés de la même manière seulement si la source lumineuse se trouve en face de l'appareil. Cette pièce a été imprimée avec une imprimante 3D. Après test, j'ai pu vérifier que l'ajout de cette pièce a été nécessaire et qu'elle isolait mieux les capteurs.

Création d'une carte

Les tests étant réalisés avec une plaquette d'essai et plein de fils, il était facile de se perdre. Donc mon tuteur m'a demandé de créer un shield pour la carte arduino afin de simplifier le montage. Pour cela j'ai utilisé le logiciel gratuit Kicad qui est l'équivalent de Proteus. C'est la première fois que quelqu'un au sein du Fablab utilisait ce logiciel du coup j'ai su suivre plusieurs tutoriels sur internet. Ensuite je leur ai laissé tous les liens que j'ai utilisés sur le serveur de l'association.

Pour commencer j'ai utilisé la partie schéma du logiciel, ce qui équivaut à ISIS. J'ai recréé le schéma électrique dans le logiciel. Vu que c'est un shield, j'ai du télécharger la bibliothèque adéquate. J'ai également changé les résistances reliées aux photorésistances pour être plus précis. En effet avec des résistances de 6,8kΩ,

le microcontrôleur a une mesure plus précise des valeurs données par les capteurs. Sur le port série elle varie entre 0 et 1000 environ, ce qui nous arrange plus qu'un intervalle entre 0 et 600.

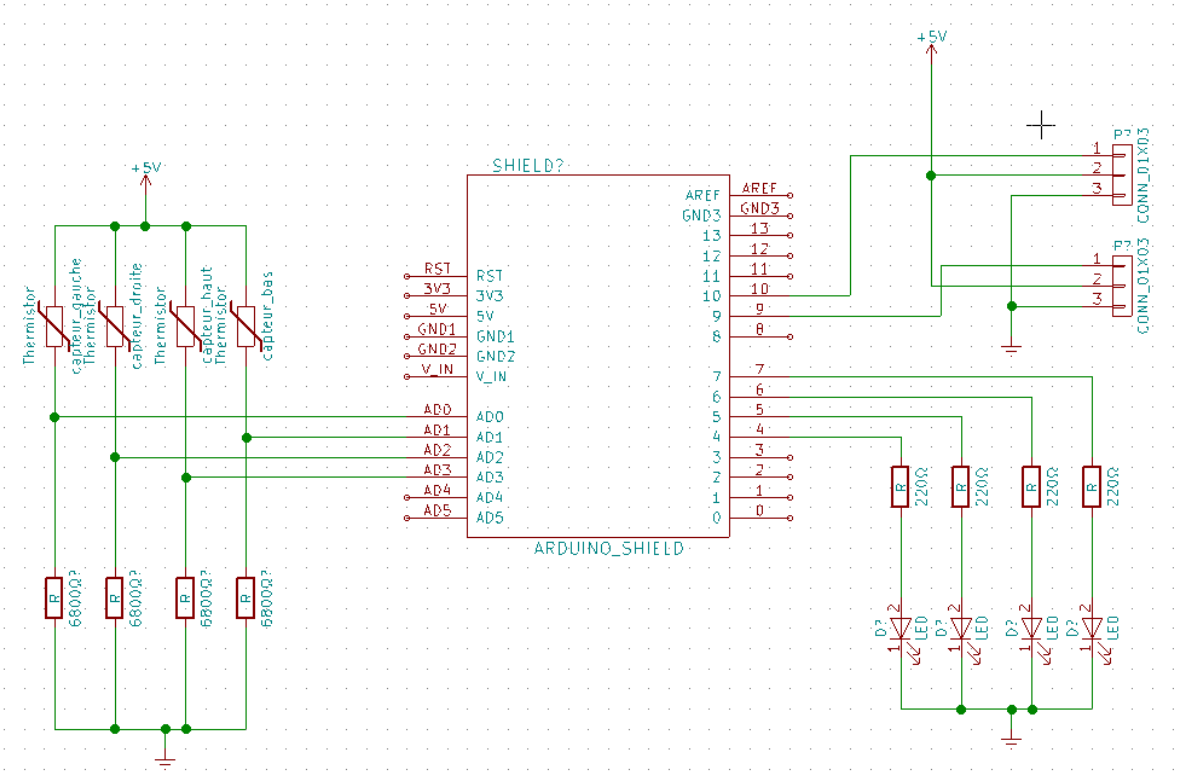

Figure 8 - Schéma électrique sous Kicad

Ensuite on demande au logiciel de faire une simulation pour vérifier que le montage fonctionne correctement. Après avoir rentré le schéma, il faut faire le rouage. Mais avant ça il faut rentrer les empreintes des composants afin qu'il puisse nous faire passer de la partie schéma à la partie routage. Une fois les empreintes rentrées, Kicad possède un équivalent d'ARES.

Dés qu'on arrive à cette étape, il faut placer les composants de manière stratégique pour que ce ne soit pas compliqué de les relier. On relie tout les composants entre eux et les broches, sauf les masses car pour cela on va utiliser un plan de masse. Un plan de masse permet de mettre toute la carte aux potentiel 0V. Cela se traduira par le fait que la carte lors de l'impression sera recouverte intégralement de cuivre.

Sur la figure ci-dessous (figure 9), le plan de masse est représenté par la couleur rouge. Les connecteurs « CONN_01X03 » et «« CONN_01X02 » représentent les connecteurs auxquels l'on va brancher les deux servomoteurs et l'alimentation 5V.

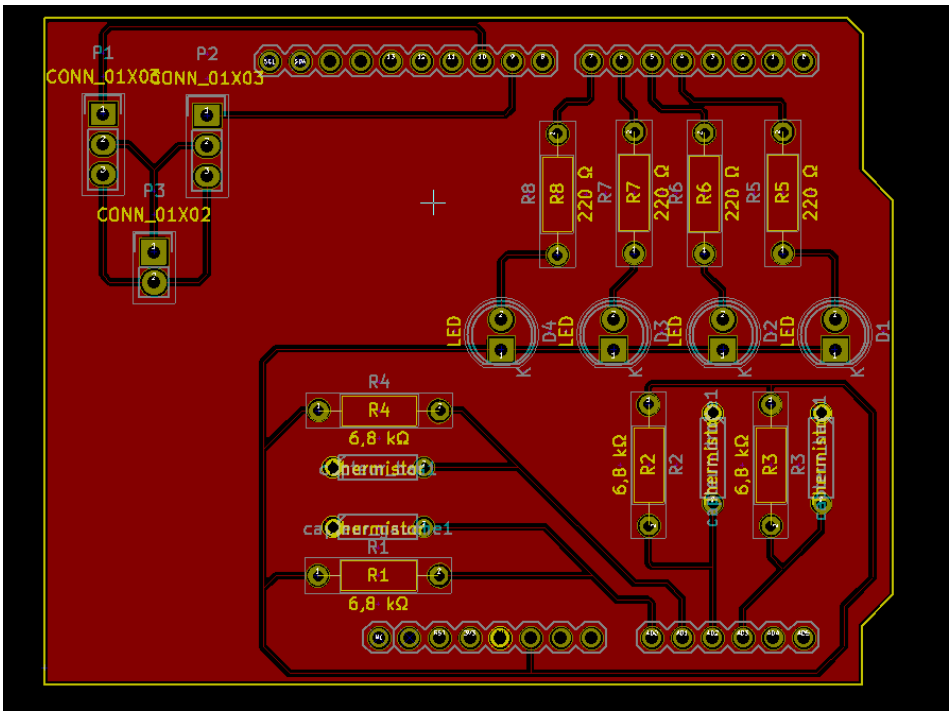

Figure 9 - Routage de la carte

Une fois le routage fait, il ne reste plus qu'à imprimer la carte. Pour cela j'ai contacté un de mes anciens professeurs lorsque j'étais à l'IUT GEII. Avec lui nous avons utilisé l'insoleuse pour imprimer la carte. Tout d'abord il faut passer la plaquette de cuivre dans une insoleuse, puis dans un liquide spécial pour nettoyer la carte. Dés que le circuit est imprimé, il faut la percer pour pouvoir souder les composants.

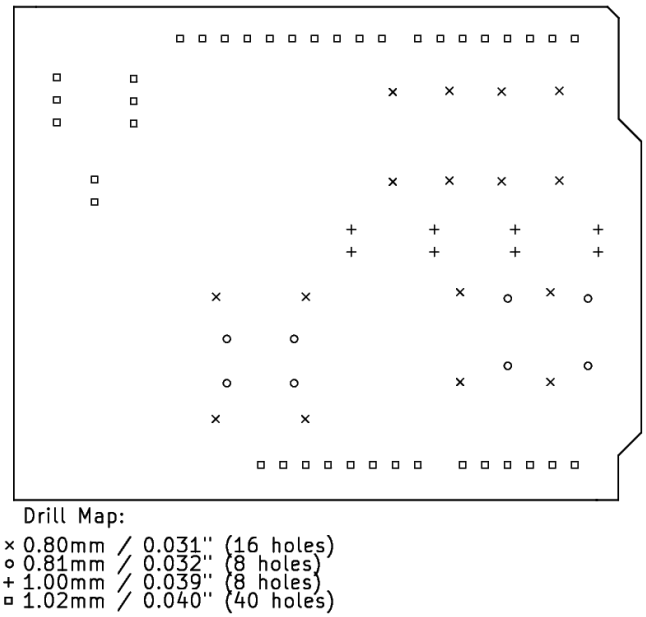

Figure 10 - plan de perçage

On soude les composants, puis on test la carte. D'abord j'ai mesuré à l'aide d'un voltmètre, si le circuit diffuse bien la tension de 5V voulue. Ensuite j'ai implanté un programme qui faisait uniquement clignoter les Leds, puis un programme qui lisait les valeurs relevées par les photorésistances. Quand tout cela a été vérifié, j'ai pu commencer la partie programmation du projet.

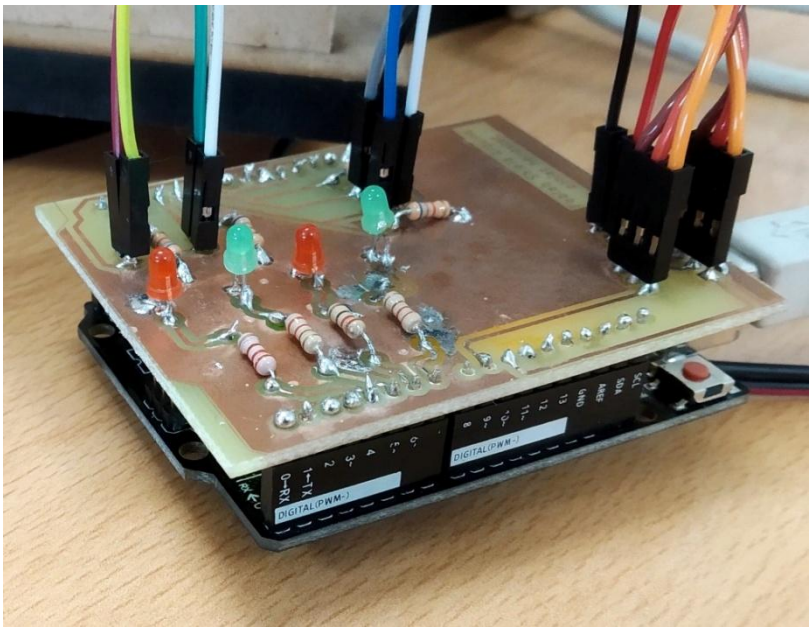

Figure 11 - Le shield arduino

Programmation de la carte

Le principe du programme est d'identifier, à une erreur tolérée près, la photorésistance qui reçoit le plus de lumière pour chaque paire (haut-bas et gauche-droite). Il est possible de modifier l'erreur afin d'obtenir un système plus ou moins précis et donc nerveux. En effet chaque photorésistance renvoie une valeur entre 0 et 1024 à la carte, pour pouvoir stabiliser l'héliostat il y a dans le programme une erreur de 30. Le microcontrôleur réalise la soustraction des valeurs entre le capteur haut et bas, et le capteur de droite et gauche.

Si les LEDs de debug ne sont pas utilisées, il n'est pas nécessaire de les retirer du programme. Le programme se décompose en plusieurs sous-fonctions :

- Asserv() constitue la fonction principale d'asservissement. Elle gère la commande des servomoteurs en fonction des valeurs renvoyées par les LDR.

- limitS() permet de limiter l'angle de commande envoyé aux servomoteurs (cf notes programmes pour les valeurs maximales tolérées). Cela permet de fixer informatiquement des butées angulaires.

- w_servo() permet d'envoyer une commande au servomoteur indiqué, en limitant automatiquement l'angle de commande.

- testLDR() permet de visualiser sur des LED les LDR qui reçoivent le plus de lumière, en incluant aussi l'erreur.

- lectureHB() et lectureGD() permettent de tracer sur deux courbes les valeurs renvoyées par les deux paires de LDR.

Objectif d'amélioration

Une fois les tests de vérification réalisés, l'héliostat fonctionnant très bien, Thierry m'a donné une piste pour améliorer le projet. Il voulait qu'à partir du traqueur solaire opérationnel, qu'un autre appareil identique puisse à partir des données collectées, refleter les rayons du soleil vers un point fixe à l'aide de miroir. Ce point fixe en l'occurrence serait un four solaire.

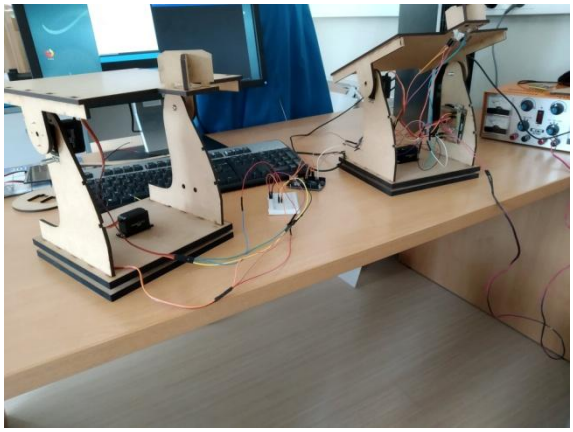

Figure 12 - Montage complet des deux appareils

Dans un premier temps, j'ai utilisé les plans que la personne avant avait réalisés pour reconstruire un second appareil. Puis j'ai commencé à faire des recherches pour savoir la manière la plus simple de faire communiquer les deux dispositifs. Chaque appareil possédant une carte arduino, j'ai décidé de les faire communiquer par une liaison I2C, en utilisant les broches A4 et A5 des microcontrôleurs.

En cherchant sur internet, j'ai trouvé le moyen d'afficher les valeurs collectées par la première carte sur le terminal série de la seconde. Mais je n'ai pas eu le temps de trouver comment récupérer ces valeurs, afin de pouvoir les utiliser pour faire marcher les servomoteurs.

Difficultés rencontrées

Durant ce stage, j'ai pu rencontrer certains problèmes, certaines difficultés techniques, problèmes de programmation.

Dans un premier temps le fait de reprendre un projet qui était en cours n'a pas été simple dés le début, en effet j'ai perdu un peu de temps pour pouvoir bien comprendre ce qui avait déjà été fait et ce vers quoi la personne avant moi voulait mener le projet.

Un autre problème, c'était le fait de ne pas avoir assez de temps pour pouvoir finir ce que Thierry m'avait dit de faire. J'ai surement arrêté de faire des recherches un peu trop rapidement pour programmer la carte et essayer de finir vite, ce qui je pense a été une erreur.

J'ai également eu quelques difficultés avec la programmation, car ce n'est pas un domaine que je maitrise particulièrement, mais avec un peu de persévérance j'ai réussi à finir de coder la première carte.

Le fait que personne ne connaissait le logiciel Kicad n'a pas été simple, car personne ne pouvait m'aider lors de débogage et j'ai du me débrouiller tout seul pour trouver les erreurs que j'avais commises.

Conclusion

A la fin du stage je peux dire que j'ai réalisé plusieurs tâches qui m'ont été demandées. J'ai réalisé un shield arduino pour simplifier le câblage, j'ai terminé la programmation de l'héliostat, laissé des liens vers de bons tutoriels Kicad, monté le nouvel appareil et commencé la programmation de celui-ci.

Ce stage a été pour moi une opportunité de voir comment fonctionne un projet et d'allier la théorie et pratique que j'ai acquise durant mes études. J'ai réalisé que pour mener à bien un projet il faut de l'organisation, essayer plusieurs méthodes pour atteindre son objectif. Aussi j'ai eu la chance d'utiliser des outils de travail que je n'avais pas l'habitude d'utiliser, le système d'exploitation Linux ou le logiciel Kicad. Donc je pense que ce stage au Fablab, même si ce n'est pas une entreprise mais une association, a été bénéfique pour moi.

J'espère que ce que j'ai réalisé, les informations que je leurs ai laissées, seront utiles et suffisantes pour la poursuite de ce projet.

Pour conclure, j'ai vraiment apprécié l'ambiance qu'il y avait au sein de l'association. Mon tuteur et les autres personnes présentes m'ont tous très bien accueilli et accompagné pendant la durée du stage.

Je tiens encore à remercier François Augereau pour ce stage et toute l'équipe du Fablab.

Index

Figure 1 - Photo d'une imprimante 3D 6
Figure 2 - Recherche du besoin 6
Figure 3 - Recherche des fonctions 7
Figure 4 - l'héliostat 10
Figure 5 - Schéma électrique du montage 11
Figure 6 - Pièce sous Freecad 11
Figure 7 - Pièce sur l'héliostat 13
Figure 8 - Schéma électrique sous Kicad 14
Figure 9 - Routage de la carte 14
Figure 10 - plan de perçage 15
Figure 11 - Le shield arduino 15
Figure 12 - Montage complet des deux appareils 16

Annexe 1

```
#include <Servo.h>
// Déclaration des servomoteurs
Servo servoBas;  // Servomoteur pilotant l'azimuth
Servo servoHaut; // Servomoteur pilotant l'elevation
////////////////////////////////
///  DECLARATION DES BROCHES  ///
////////////////////////////////
// PIN LDR (Sens: vue de face)
int LDR_HAUT = A3;
int LDR_BAS = A1;
int LDR_GAUCHE = A0;
int LDR_DROITE = A2;
//PIN LEDs
int LED_H = 7;
int LED_B = 4;
int LED_G = 6;
int LED_D = 5;
// PIN SERVOMOTEURS
int pin_servoBas = 9;
int pin_servoHaut = 10;
////////////////////////////////
/// DECLARATION DES VARIABLES ///
////////////////////////////////
// VARIABLES DE LECTURE DES LDRs
int LDR_H=0;
int LDR_B=0;
int LDR_G=0;
int LDR_D=0;
// Variables de calculs
int erreur = 30; // Erreur toleree entre les valeurs renvoyees par les photoresistances. Plus cette erreur est petite, plus le système sera nerveux (et risque d'être instable)
// Programme d'initialisation. S'execute une seule fois, en début de programme
void setup() {
  // Les servomoteurs sont déclarés et une premiere valeur leur est donnee, afin d'eviter une initialisation trop brutale
  servoBas.attach(pin_servoBas);  // Attache le servo du bas
  servoBas.write(90);
  servoHaut.attach(pin_servoHaut); // Attache le servo du haut
  servoHaut.write(160);
  Serial.begin(9600); // Lance la communication serie a 9600 bits/s
/// DECLARATION DES ENTREES ET SORTIES ///
  // Les LDR sont declarees comme des entrees:
```

```
  pinMode(LDR_HAUT, INPUT);
  pinMode(LDR_BAS, INPUT);
  pinMode(LDR_GAUCHE, INPUT);
  pinMode(LDR_DROITE, INPUT);
  // Les LEDs sont declarees comme des sorties:
  pinMode(LED_H, OUTPUT);
  pinMode(LED_B, OUTPUT);
  pinMode(LED_D, OUTPUT);
  pinMode(LED_G, OUTPUT);
}
////////////////////////////////
///      SOUS-FONCTIONS     ///
////////////////////////////////
// Fonction permettant de piloter un servo a partir d'une valeur donnee dans le programme
void w_servo(int servo, int val) {
  switch(servo) { // Selection du servomoteur
    case 0:
      servoBas.write(limitS(0,val));
      break;
    case 1:
      servoHaut.write(limitS(1,val));
      break;
  }
}
// Fonction test des LDR (but: verifier le bon fonctionnement des LDR et fixer une erreur), allume les LEDs associees aux
capteurs
void testLDR(){
  //Lecture des valeurs renvoyees par les LDR:
  LDR_H=analogRead(LDR_HAUT);
  LDR_B=analogRead(LDR_BAS);
  LDR_G=analogRead(LDR_GAUCHE);
  LDR_D=analogRead(LDR_DROITE);
 // Paire haut-bas
 if (abs(LDR_H-LDR_B) > erreur) {   // Si l'ecart entre les 2 LDR est significatif, on allume la LED correspondant a la LDR la
plus eclairee
  if (LDR_H > LDR_B) {
    digitalWrite(LED_H, HIGH);
    digitalWrite(LED_B, LOW);
  }
  else{
    digitalWrite(LED_H, LOW);
    digitalWrite(LED_B, HIGH);
  }
 }
```

```
 else {   // Sinon, on éteint les LEDs
  digitalWrite(LED_H, LOW);
  digitalWrite(LED_B, LOW);
 }

 // Paire gauche-droite
 if (abs(LDR_G-LDR_D) > erreur) {  // Si l'ecart entre les 2 LDR est significatif, on allume la LED correspondant a la LDR la
plus eclairee
  if (LDR_G > LDR_D) {
    digitalWrite(LED_G, HIGH);
    digitalWrite(LED_D, LOW);
  }
  else{
    digitalWrite(LED_G, LOW);
    digitalWrite(LED_D, HIGH);
  }
 }
 else {   // Sinon, on éteint les LEDs
  digitalWrite(LED_G, LOW);
  digitalWrite(LED_D, LOW);
 }
}
// Asservissement en position des servomoteurs
void asserv() {
  int angleActuel0;
  int angleNouveau0=servoBas.read();
  int angleActuel1;
  int angleNouveau1=servoHaut.read();
  //Lecture des valeurs renvoyees par les LDR:
  LDR_H=analogRead(LDR_HAUT);
  LDR_B=analogRead(LDR_BAS);
  LDR_G=analogRead(LDR_GAUCHE);
  LDR_D=analogRead(LDR_DROITE);
if(LDR_H < 400 && LDR_B < 400 && LDR_G < 400 && LDR_D < 400){ // cette boucle permet d'effectuer rotation
uniquement s'il fait jour
   w_servo(0,10);
   w_servo(1,160); // s'il fait nuit, le robot se met en position pour attendre le soleil le lendemain matin
}
else{
  // Paire haut-bas
 if (abs(LDR_H-LDR_B) > erreur) {
  // Si c'est plus lumineux en haut, il faut relever le panneau
  if (LDR_H > LDR_B) {
   angleActuel1 =  servoHaut.read();
```

```
  angleNouveau1 = angleActuel1-1;    // Decrementer l'angle de commande = relever le panneau
  }
  else{
  angleActuel1 = servoHaut.read();
  angleNouveau1 = angleActuel1+1;    // Incrementer l'angle = baisser le panneau
  }
 }
 w_servo(1,angleNouveau1);  // Pilotage du servomoteur.
  // Paire gauche-droite
  if (abs(LDR_G-LDR_D) > erreur) {
    // Si c'est plus lumineux a gauche, il faut tourner le panneau vers la "droite"
    if (LDR_G > LDR_D) {
     angleActuel0 = servoBas.read();
     angleNouveau0 = angleActuel0-1;
    }
    else{
     angleActuel0 = servoBas.read();
     angleNouveau0 = angleActuel0+1;
    }
   }
   w_servo(0,angleNouveau0);
delay(50);
}
}
// Fonction de limite d'angle pour les servomoteurs
// Permet de fixer des butees angulaires et d'empêcher des collisions ou blocage/forçage mecanique
int limitS(int i, int angle) {  // i=0 pour le servo du bas, i=1 pour le servo du haut
  int angle2 = angle; // Nouvel angle de sortie
  // Declaration de variables qui seront les limites angulaires du servomoteur (en degres)
  int limBasse;
  int limHaute;
  // Definition des limites angulaires selon le servomoteur (en degrés)
  switch(i) {
    case 0: // Servomoteur du bas
      limBasse = 5;
      limHaute = 175;
      break;
    case 1: // Servomoteur du haut
      limBasse = 90;
      limHaute = 175;
      break;
    default: break;
  }
  // Si l'angle demande au servo depasse les limites, on fixe cet angle a la limite depassee
```

```
  if (angle > limHaute) {
    angle2 = limHaute;
  }
   if (angle < limBasse) {
    angle2 = limBasse;
  }
return angle2;
}
// FONCTIONS DE DEBUG
// Lecture et ecriture des valeurs renvoyees par les LED haut et bas
void lectureHB() {
  LDR_H = analogRead(LDR_HAUT);
  LDR_B = analogRead(LDR_BAS);
   Serial.print("          , ");
  Serial.print(LDR_H);
  Serial.print("   , ");
  Serial.print(LDR_B);
  Serial.println();
  delay(100);
}
// Lecture et ecriture des valeurs renvoyees par les LDR gauche et droite
void lectureGD() {
  LDR_G = analogRead(LDR_GAUCHE);
  LDR_D = analogRead(LDR_DROITE);
  Serial.print(LDR_G);
  Serial.print(", ");
  Serial.print(LDR_D);
  Serial.println();
}
////////////////////////////////
///   FONCTION PRINCIPALE    ///
////////////////////////////////
void loop() {
  asserv();
  //lectureGD();
  testLDR();
 // lectureHB();
}
```