03/02/23

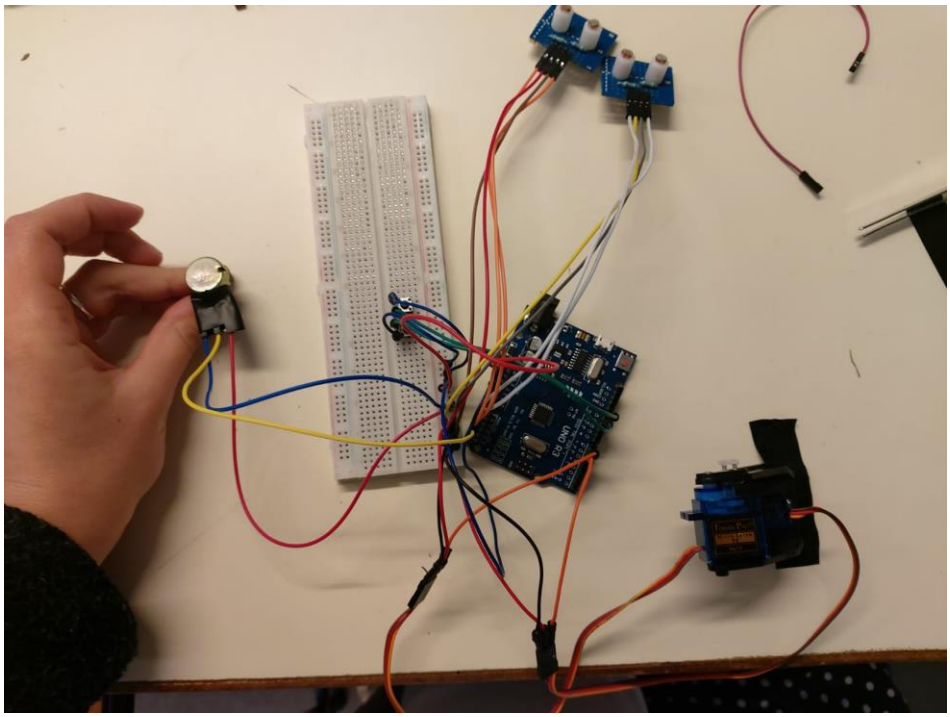
Delafosse Aglaé

Rapport de séance 5

Durant la première partie de séance j'ai recherché comment récupérer l'énergie du panneau solaire pour recharger une batterie portable. Après avoir regardé plusieurs options différentes j'ai opté pour un convertisseur avec une entrée 6V et une entrée 12V, qui ressortait l'énergie avec une tension de 5V, adaptée donc à la charge d'un appareil électronique. J'ai appelé un professeur pour valider mon choix et envoyé le lien du modèle à M. Masson pour qu'il puisse l'acheter.

Durant la seconde partie j'ai essayé d'ajouter la fonction de « réglage manuel » à notre solar Tracker. Pour ça j'ai décidé d'ajouter à mon montage deux boutons poussoirs et un potentiomètre. Il y a un bouton pour la rotation haut-bas (entrée digitale 4) et un bouton pour la rotation gauche-droite (entrée digitale 3). Le potentiomètre (entrée analogique 5) permet de régler l'intensité de rotation voulue. Les deux boutons agiraient comme des interrupteurs : lorsque l'un d'eux est actionné le programme le détecte et le solar tracker passe en mode manuel.

Après avoir trouvé quelques exemples de conception sur internet je me suis lancée dans le montage complet.



Et j'ai réalisé les premières modifications en conséquence sur mon code. J'ai créé une fonction « automaticTracker » hors du loop qui sera utilisée lorsqu'aucun des boutons n'est actionné, et une fonction « manualTracker » qui sera utilisée sinon.

Mon programme ne fonctionnant pas j'ai fait des recherches pour me rafraîchir la mémoire sur la façon de procéder pour transformer un bouton poussoir en interrupteur mais je n'ai pas eu le temps de faire les modifications pendant la séance.

```
#include <Servo.h>

int haut;
int gauche;
int droite;
int bas;
int mode;
int ButtonState1;
int PrevButtonState1;
int ButtonState2;
int PrevButtonState2;
int state;
int prevState;
int ButtonUD=12;
int ButtonLR=11;
int reglage=4;  //potentiomètre
int LDR1=0;   //gauche
int LDR2=1;   //droite
int LDR3=2;   //haut
int LDR4=3;   //bas
Servo servoUD;
Servo servoLR;

void setup() {
  pinMode(ButtonUD,INPUT);
  pinMode(ButtonLR,INPUT);
  pinMode(reglage, INPUT);
  PrevButtonState1=digitalRead(ButtonUD);
  PrevButtonState2=digitalRead(ButtonLR);
  prevState=analogRead(reglage);
  servoUD.attach(5);
  servoLR.attach(6);
  Serial.begin(9600);
```

```
void loop() {
  // put your main code here, to run repeatedly:
  ButtonState1=digitalRead(ButtonUD);
  ButtonState2=digitalRead(ButtonLR);
  Serial.println(ButtonState1);
  Serial.println(PrevButtonState1);

  if (ButtonState1!=PrevButtonState1||ButtonState2!=PrevButtonState2){
      if(ButtonState1!=PrevButtonState1){
        mode=1;
        manualTracker();
      }
    }
   else{
    automaticTracker();
   }
}

void manualTracker(){
  state=analogRead(reglage);
  if (mode==1){
    if (state<prevState){
      servoUD.write(servoUD.read()-1);
    }
    if (state>prevState){
      servoUD.write(servoUD.read()+1);
    }
  }
  else{
    if (state<prevState){
      servoLR.write(servoLR.read()-1);
    }
    if (state>prevState){
      servoLR.write(servoLR.read()+1);
```

```
void automaticTracker(){
  gauche=analogRead(LDR1);
  droite=analogRead(LDR2);
  haut=analogRead(LDR3);
  bas=analogRead(LDR4);
  gauche=map(gauche,1,959,0,180);  //étalonnage fait
  droite=map(droite,0,977,0,180);  //étalonnage fait
  haut=map(haut,2,967,0,180);   //en attente de l'étalonnage
  bas=map(bas,0,978,0,180);   //en attente de l'étalonnage
  int diffHB=haut-bas;
  int diffGD=gauche-droite;

  if (abs(diffHB)>2){
    if (diffHB<0){
      if (servoUD.read()<180){
        servoUD.write(servoUD.read()+1);
      }
    }
    else{
      if (servoUD.read()>0){
        servoUD.write(servoUD.read()-1);
      }
    }
  }
  if (abs(diffGD)>2){
    if (diffGD<0){
      if (servoLR.read()<180){
        servoLR.write(servoLR.read()+1);
      }
    }
    else{
      if (servoLR.read()>0){
        servoLR.write(servoLR.read()-1);
      }
    }
  }
  delay(100);
}
```