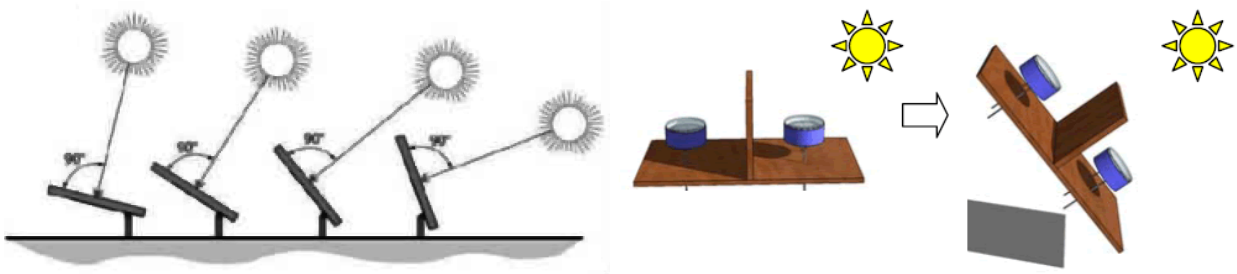
C5-160551-INFO
Objets Connectés :
Programmation Microcontrôleurs - TP5

année universitaire 2021-2022

1 TP type “cahier des charges”

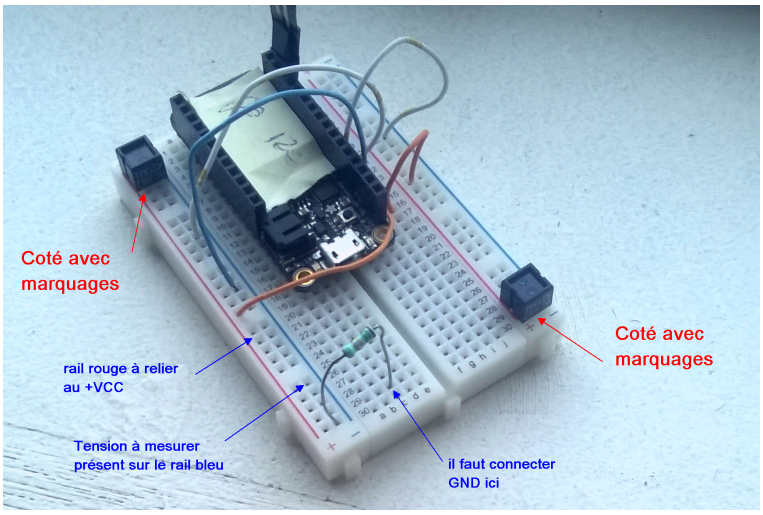
Le but de ce TP est de développer un logiciel qui, associé à une mécanique motorisée et équipée de capteurs, doit permettre une orientation optimale (en inclinaison) d’un panneau solaire (eau chaude ou photovoltaique). En effet, le rendement d’un panneau solaire est maximal quand le panneau est perpendiculaire aux rayons du soleil.
Pour cela, nous disposons de deux capteurs de luminosité de part et d’autre du panneau (voir schéma) et l’orientation optimale est obtenu lorsque les deux capteurs perçoivent la même luminosité. Si le capteur du haut perçoit une luminosité plus importante, il faut modifier l’angle et “coucher” le panneau un peu plus, ou le redresser lorsque le capteur du bas perçoit plus de luminosité, et ce réglage doit se faire en continu (disons une mesure et un ajustement — si nécéssaire — toutes les secondes). On appelle cela un asservissement.



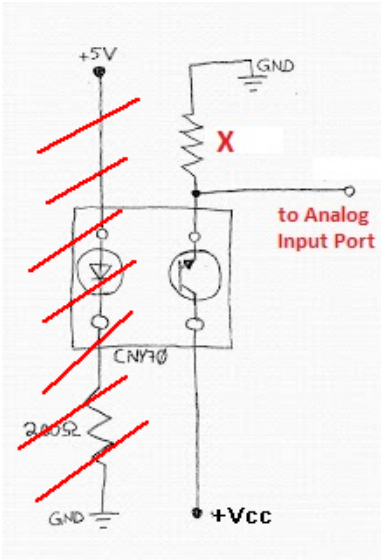
2 Câblage et tests

1. Disposez votre platine microcontrôleur et les deux capteurs de luminosité plus ou moins comme illustré sur les photos. Sur la photo, le connecteur est orienté vers le milieu de la plaque de test : en effet, la plaque sera placée sur un support mobile et il vaut mieux que seul la partie souple du câble USB dépasse. Avec une petite plaquette de montage, il vaut mieux placer les photo-transistors sur les rails qui servent généralement pour les lignes d’alimentation, en les plaçant aux extrémités opposés de la plaque d’essai (mais ne vous trompez pas dans l’orientation du capteur et le câblage). De façon générale, tenter de séparer le plus possible les deux photo-transistors.

   Ne pas se fier au câblage représenté sur la photo, tenir compte des légendes, du placement de l’ESP32, des photo-transistors, de la résistance (présente ici que pour un seul phototransitor)



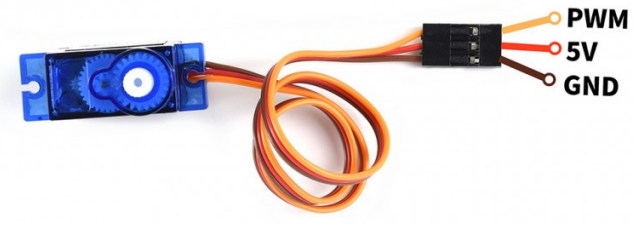
2.1 Câblage et test des capteurs de luminosité



Nous utiliserons deux capteur CNY70, dont seul la partie phototransistor sera cablée (voir TP4). Utilisez deux résistances identiques pour le capteur 1 et pour le capteur 2. Reliez la sortie (tension au borne de la resistance X) du capteur 1 sur l'entrée analogique A3 et la tension du capteur 2 sur A4.
Réalisez le montage suivant (pour chaque photo-transistor). Pour la résistance X, prendre une valeur entre 100 KOhms et 200 KOhms, mais les deux résistances doivent être identiques.
Testez en représentant le code de test du TP4 (en le modifiant un peu en mettant les bonnes valeurs) et passez avec la main au dessus des capteurs.

2.2 Test du servo-moteur

Trouvez une bibliothèque adaptée (n'oubliez pas que votre processeur est un ESP32, et pas une platine arduino de base). Vous pouvez utiliser internet ou bien la fonction de recherche intégrée dans le gestionnaire de bibliothèque de l'IDE Arduino (Tools → Manage Librairies). L'installation de la bibliothèque se fait en utilisant le gestionnaire de bibliothèque. Après avoir installé la bibliothèque, consultez la liste des exemples (File → examples), il y aura probablement un programme de test ou d'exemple qui sera mis à disposition pour illustrer l'usage de la bibliothèque).



Branchez la ligne de commande (PWM ou signal) de votre servomoteur sur la broche 25 (ATTENTION : Sur le shema montrant le brochage de la platine Adafruit Huzzah 32, il y permutation entre les broches 25 et 26). Le 5V est

disponible sur vos platines ESP32 sur la broche marquée USB. Connectez la masse (GND).

Tester le fonctionnement du servomoteur dans la partie setup() de votre code, par exemple mettre le servomoteur en position 0°, pause de 1 seconde, puis 180°, pause de 1 seconde, puis 90°.

2.3 Utilisation de 2 touches tactiles et tests

Brancher un simple fil sur le GPIO 4, fil que l'on laisse "'en l'air"'. Cette broche peut être configurée comme entrée capacitive nommé T0 (Touch 0), déjà défini dans le framework. Avec le code suivant on lira une valeur qui change lorsqu'on touche le fil (à tester dans loop() pour avoir une lecture continue) :

```
Serial.println(touchRead(T0));  // get value using T0
delay(500);
```

```
T0 (GPIO 4)
T1 (GPIO 0)
T2 (GPIO 2)
T3 (GPIO 15)
T4 (GPIO 13)
T5 (GPIO 12)
T6 (GPIO 14)
T7 (GPIO 27)
T8 (GPIO 33)
T9 (GPIO 32)
```

Vérifiez l'évolution selon que vous touchez le fil.
Il faut deux touches capacitives qui serviront au mode manuel, vous devez donc branchez deux fils que vous pouvez toucher. Vous devez déterminer un seuil qui indiquera si oui ou non il y a un doigt qui touche le fil.
Voir `www.youtube.com/watch?v=4YY7TutRrQE`

3 Mode manuel et déplacement du servomoteur

Apres demarrage et initialisation, en touchant l'une des touches capacitives, vous devez faire bouger le servomoteur dans un sens, et le servomoteur doit bouger dans l'autre sens en agissant sur l'autre fil.

Profitez de cette étape pour identifier les valeurs de "'fin de course"' pour votre système articulé qui supportera la platine. L'utilisation des touches tactiles autorise également un mode interruptif, voir internet, avec la fonction :
`touchAttachInterrupt(T, callback, threshold);`

4 Mode automatique

Pour l'instant, une variable booléenne permettra de distinguer entre le mode manuel et le mode automatique. La LED LED_BUILTIN doit s'allumer pour indiquer que vous êtes en mode automatique.

Dès qu'il y a action sur une touche tactile, on doit passer en mode manuel, et le servomoteur doit positionner le support articulée en position médiane (à l'horizontale]. Le passage en mode automatique s'effectue via l'interface HTML et le smartphone. Lorsqu'on passe en mode automatique, l'asservissement commence de suite.

5 Développement de l'algorithme d'asservissement

Dans le mode automatique, il faut faire des mesures (disons 2x par seconde) et ajuster le déplacement du support articulé pour l'orienter en regard de la source lumineuse.

Vous pouvez commencer la mise au point sans fixer la platine sur le support, c'est plus facile.
Voici quelques éléments pour développer votre algorithme :

- Mesurer la tension au niveau de chaque photo-transistors (par ex. val1 et val2). Comme les composants électroniques ont une tolérance de fabrication, **même avec un éclairage identique, les valeurs fournis par les photo-transistors ne seront pas identiques**. A vous de voir si vous devez corriger ces valeurs et de quelle façon !
- L'écart entre val1 et val2 est mesuré, et c'est cet écart doit piloter (en passant probablement par des ajustement) le servomoteur pour pointer en direction de la source lumineuse.
- Lorsque vous envoyez un ordre de positionnement vers le servomoteur, il faut laisser à celui-ci le temps de se positionner (donc, mettre un delay(200) après un ordre de positionnement, c'est assez logique.
- Mettre votre algorithme au point sans utiliser les modes de veille. Le cas echéant, vous pouvez rajouter un mode light sleep lorsque tout fonctionne ?.

6 Mise en mode automatique

Seul une commande à distance permet de repasser en mode automatique. Pour cela, votre ESP32 emmétra un hotspot (pour éviter les conflits dans le nommage des réseaux WiFi, le SSID de votre réseau sera le nom d'un des membre du binôme, en MAJUSCULES). Cherchez "'ESP32 ACCESS POINT"' pour avoir des infos.
Vous vous connecterez sur ce réseau wifi avec votre smartphone. Avec un navigateur, vous naviguerez vers la page d'accueil (le serveur se trouve à l'adresse par defaut 192.168.4.1). Le serveur devra transmettre une page HTML comportant un seul lien de navigation (éventuellement présenté comme un bouton) qui devra placer votre système en mode automatique.
Voir code suivant, reprenant une partie du code de cette page : `emery.claude.free.fr/esp32-serveur-web-simple.html`
Attention, pas de création d'ACCESS POINT sur la page citée, mais c'est complété dans le code ci-dessous.

```
#include <WiFi.h>
#include <WebServer.h>

const char* ssid = "VOTRE_SSID"; // Mettre le ssid de votre réseau Wifi
const char* password = "VOTRE_MOT_DE_PASSE"; // Mettre le mot de passe de votre réseau Wifi

WebServer server(80); // Creation dun objet de type serveur HTTP

void handleRoot(){ // Callback declenché en cas de navigation vers la page dacceuil (root)
// Syntaxe décriture pour être compatible avec le C++ / Arduino
// String page = " xxxxxxxx ";
// page += " xxxxx ";
// etc ...
String page = "<!DOCTYPE html>"; // Début page HTML
page += "<head>";
page += " <title>Serveur ESP32</title>";
page += " <meta http-equiv=refresh name=viewport content=width=device-width, initial-scale=1 charset=UTF-8/>";
page += "</head>";
page += "<body lang=fr>";
page += " <h1>Mon Serveur</h1>";
page += " <p>Ce serveur est hébergé sur un ESP32</p>";
page += " <a href=mode_auto>Cliquez ici pour passer en mode automatique</a>";
page += "</body>";
page += "</html>"; // Fin page HTML
server.send(200, "text/html", page); // Envoi de la page HTML
}

void handleNotFound(){ // Callback declenché en cas de navigation vers une cible inconnue
server.send(404, "text/plain","404: Not found");
}

void setup() {
Serial.begin(115200);
pinMode(LED_BUILTIN, OUTPUT);
ma_pause = 1000;
delay(1000);
Serial.print("Setting AP (Access Point)...");
//WiFi.softAP(ssid, password); // Remove the password parameter, if you want the AP (Access Point) to be open
WiFi.softAP(ssid);
IPAddress IP = WiFi.softAPIP();
Serial.print("AP IP address: ");
Serial.println(IP);
server.on("/", handleRoot); // si navigation vers (page daccueil) appel du callback handleRoot
server.onNotFound(handleNotFound); // si navigation vers cible inconnue, appel du callback handlenotFound
server.begin(); // Initialisation du serveur web
Serial.println("Serveur web actif");
}

void loop() {
server.handleClient(); // Attente de demande du client
// votre code
}
```