Documentation MOOREV-Timer-Camera ScFo

Information:

Groupe de projet: L1 - Science Formelles Maths-Info - (février - avril 2023)

- CHU Amélie amelie.chu@etu.sorbonne-universite.fr

- SAE LIM Thierry thierry.sae_lim@etu.sorbonne-universite.fr

- PERRIN DE BRICHAMBAUT Jules jules.perrin_de_brichambaut@etu.sorbonne-universite.fr

Contexte:

Dans le cadre de notre projet d'ARE, nous nous focalisons sur la récolte de données images et vidéos d'espèces marines à l'aide de caméras et de capteurs.

Objectif:

Notre objectif est de concevoir un système autonome qui permet d'enregistrer ces données en profondeur sur une longue durée (plusieurs semaines à plusieurs mois), à faible coût et à faible consommation d'énergie.

Le but de ce tuto est de mettre en place un système autonome sous marin qui permet de prendre des données photos et vidéos d’espèces aquatiques sur une durée déterminée. Notre objectif est donc de finaliser un prototype peu coûteux et ayant une consommation d’énergie minimale.

Matériel:

- GoPro Hero 3+S

- Adafruit ESP32 Feather [0]

- Arduino NANO

- Breadboard

- connecteur arrière de la caméra (connecteur herobus) modèle DD1P030MA1

- Carte SD 64 Go (ou 32 Go)

- ESP32-Cam

- Multimètre (modèle utilisé TENMA 72-14620)

Le tuto se découpera en plusieurs parties:

1. IDE et programmation des microcontrôleurs
2. Réalisation de tests de consommations
3. Script de la GoPro
4. Assemblage du prototype
5. Alternative: ESP32-Cam
6. Références

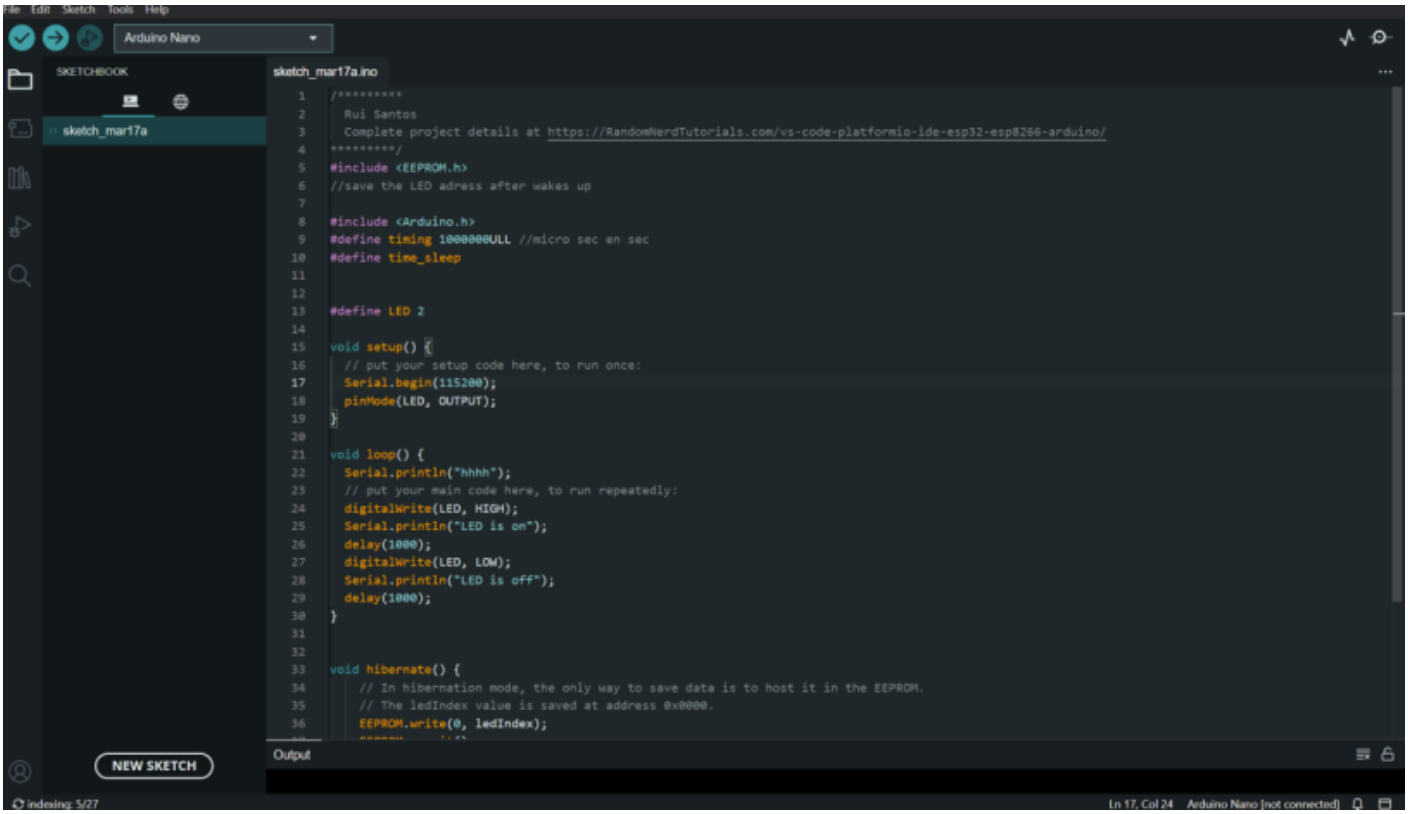
Machines utilisées:

- fer à souder

1.TUTO: IDE et programmation des microcontrôleurs

Etape 1 :

On installe tout d’abord l’IDE Arduino (Integrated Development Environment) pour pouvoir programmer les scripts des deux cartes: https://www.arduino.cc/en/software [1] (lien pour le téléchargement).



Interface du logiciel en question

Une fois téléchargé, lancez l'application. En haut à gauche, choisir le microcontrôleur adapté (dans notre cas, il s'agit d'un Adrafruit ESP32 Feather, d'un ESP32-CAM ou d'un Arduino Nano). Sur la gauche, cliquer sur la deuxième icône 'BOARDS MANAGER' et rechercher 'esp32'. Installer la plus récente version de la librairie 'esp32' par Expressif Systems. Cela permettra d'avoir la gestion pour programmer les ESP32.

Etape 2:

Nous allons maintenant écrire les scripts pour les deux cartes. Notre script va simuler une séquence où la carte allume une LED toutes les minutes pendant 5 secondes.

Le lien vers notre GitHub contient les scripts des deux cartes dans les deux fichiers ESP32 et Arduino [2] :

https://github.com/ursusnocte/ARE_Curious_Info-Moorev-Timer-Camera [3]

Pour transmettre les fichiers, relier la carte à l'ordinateur, sélectionner le port COM de la carte, et cliquer en haut à gauche sur le bouton flèche 'upload'. Pour chaque modification du script, il faudra réitérer cette opération.

2.TUTO: Réalisation de tests de consommations

On réalise maintenant les tests de consommations. Pour cela, nous allons utiliser le multimètre. Nous souhaitons récolter les données.

Etape 1: installation du logiciel

En fonction du multimètre utilisé, la fonctionnalité peut ne pas être disponible. Il faudra alors noter manuellement les données. Dans le cas contraire, installer le logiciel associé. Dans notre cas avec ce modèle, installer le logiciel à l'aide du CD fourni avec le manuel d'utilisation. Lancer le logiciel puis brancher l'ordinateur au multimètre.

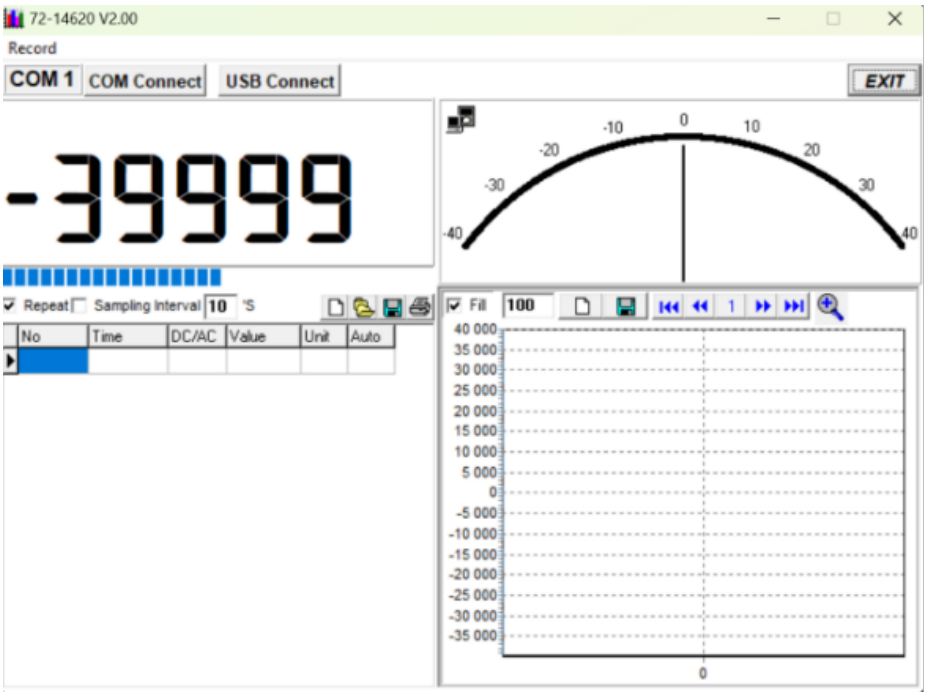
Modèle du multimètre utilisé dans notre cas : TENMA 72-14620

- Installer sur l'ordinateur le modèle du multimètre correspondant au modèle

- Connecter l'ordinateur et l'appareil grâce au câble fourni avec l'appareil

- Régler le logiciel sur la connexion via USB, l'enregistrement est prêt à démarrer avec le bouton «Démarrer».

- Différentes options sont disponibles une fois l'enregistrement terminé: enregistrer les données dans un fichier ou les imprimer.



Inteface du logiciel

Etape 2: Branchement

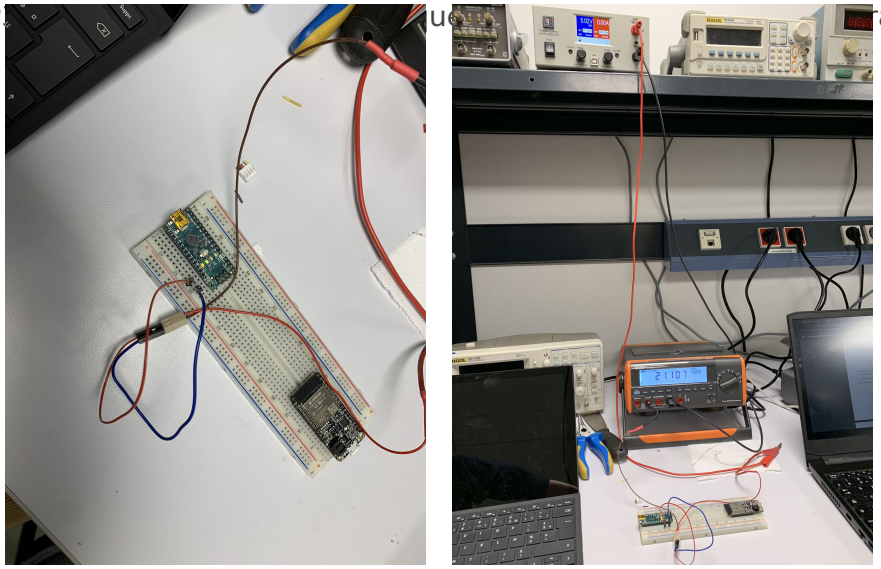
Nous allons réaliser un branchement pour alimenter les cartes et mesurer les consommations. On enfiche les cartes sur une breadboard, puis on les alimente avec un générateur réglé sur 5V. On n'oublie pas le multimètre dans le branchement.

On alimente les cartes et on récolte les données sur une durée de 3 séquences, soit environ 3 minutes afin d'avoir une moyenne à la fin. On enregistre le fichier sous forme Excel (ou autre format).

Afin de d'alimenter les cartes pour réaliser les mesures de l'intensité utilisé, on connecte une source en +5V sur le pin 5V et la masse sur le pin GND de l'arduino Nano [4].

Pour l'ESP32 (avec une carte Adafruit HUZZAH32 - ESP32 Feather), connecter la broche USB au 5V et la masse à la broche GND [5].

Ne pas oublier de brancher le multimètre en série du microcontrôleur et de ne pas utiliser le port USB des cartes arduino ou ESP (sauf si le multimètre est branché en USB) !



Branchements et circuit réalisés

On récolte les données de consommation pour les deux cartes à l'aide du logiciel sur ordinateur.

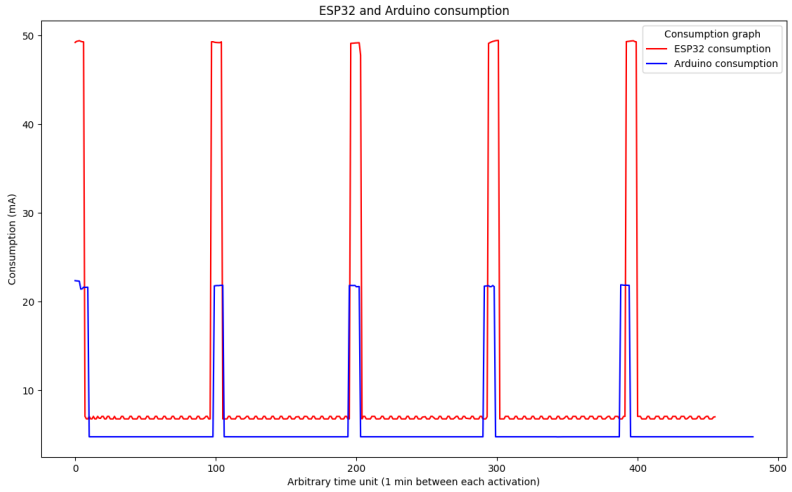
Etape 3: analyse des données

Nous analysons ensuite les données en utilisant les bibliothèques python Pandas et Matplotlib avec Python.

Pour pouvoir exploiter le fichier Excel sous Python, il ne faut pas oublier de remplacer les «,»

par des «.».

Le fichier qui nous a permis de faire le graphique est disponible sur le GitHub sous le nom de «plot_conso» [3].



On observe que l'Arduino Nano consomme moins que l'ESP32. Les pics de chaque lignes correspondent à l'allumage de la LED de chaque carte, et l'espace entre chaque pic correspond au mode veille. L'Arduino semble le mieux adapté en terme de consommation.

	Arduino	**ESP32**
mean	6.1727 mA	10.5101 mA
standard deviation	4.6997	11.8473
total time	5.0 min 18.0 s	5.0 min 0.0 s
total time consumption	2981.44 mA	4792.6 mA
average consumption	22.2219 mW/h	37.8363 mW/h

Etape 4: soudure des fils

Nous avons choisi de contrôler la caméra via le connecteur arrière [6] qui permet d'avoir accès à une grande partie des fonctionnalités. Pour utiliser notre connecteur, il nous faut souder des fils dessus.

Les 30 fils de couleurs différentes afin que l'on puisse distinguer les pins adjacentes

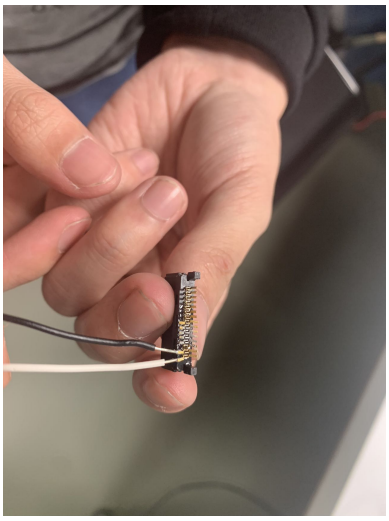
Les pins du connecteur

Nous allons souder les 30 pins. Il est possible pour plus de facilité de ne souder que les pins concernés, mais le tuto se reposera sur les 30 pins [7].

- Couper et dénuder 30 fils

- Chauffer les fers à souder à 350° (pas trop sinon l'étain fond mal)

- Prendre le fer à souder et plaquer fermement le fil au pin, puis approcher l'étain pour le faire fondre grâce au flux.



Brasure de deux pins adjacentes

3.TUTO: Script de la GoPro

La GoPro Hero 3+S a pour avantage de pouvoir lancer un script présent sur la carte SD dès son allumage. Nous allons configurer la caméra et écrire le script.

Etape 1:

Allumer la caméra et se rendre dans les paramètres. Chercher l'option «One Button» et la configurer sur OFF.

Configurer au passage la résolution souhaitée pour les photos et vidéos

Etape 2:

Pour le script de la GoPro, il faut enregistrer le fichier sous extension ".ash" . Il est disponible sur notre GitHub [4]. Le script que nous avons écrit permet de prendre une vidéo et trois photos avant de redémarrer. Disposer de ce script dans la racine de la carte SD/créer un dossier AUTOEXEC qui contiendra le fichier [8]. Enregistrer le fichier sous le nom «autoexec.ash»

Commandes	Utilisation
sleep x	Rendre la caméra inactive pendant x secondes. Temps d'attente
t app appmode video	Passage en mode vidéo de la caméra
t app appmode photo	Passage en mode photo
t app button shutter PR	Enclenchement du bouton déclencheur pour prendre une photo, prendre une vidéo/arrêter une vidéo

Il est intéressant de programmer les séquences en tenant en compte de la batterie et de la capacité de stockage de la carte SD. Pour la GoPro Hero3+S, nous disposons de 64G et d'une

batterie d’environ 10 000A. En prenant en compte la taille d’une photo (12MPX) et d’une seconde de vidéo (en 1080p), on peut calculer combien de temps peut tenir notre prototype.

Taille en ko	1 photo	vidéo (1 sec)
GoPro Hero 3+S	4000	5000

On calcule ainsi par approximation le nombre de jour que nous pouvons tenir sous l’eau tout en enregistrant des données avec la formule:

(Capacité de la carte SD) / ((nombre de photos par jour)*(taille d’une photo) + (temps de vidéo en seconde par jour)*(taille d’une seconde de vidéo))

Le calcul peut se faire sur une durée d’un mois ou bien d’un an.

Par exemple, pour 10 secondes de vidéos et 24 photos par jour, cela nous fait:

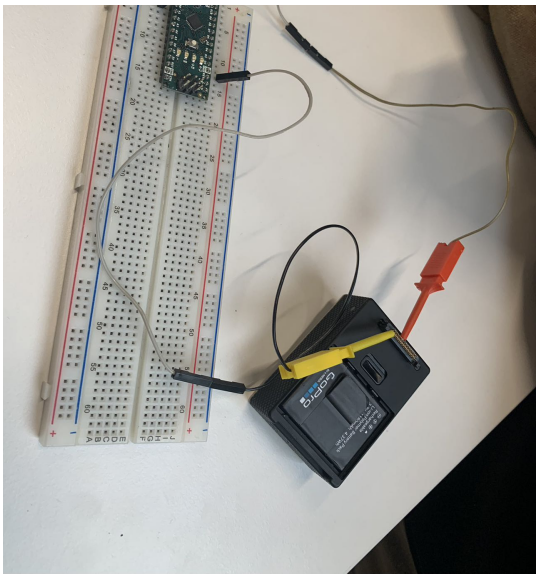
64000000/(24*40000 + 10*5000) = 438,36 soit environ 438 jours, équivalent à environ 14 mois.

Il est aussi possible de calculer la durée en fonction de la consommation de la caméra [9].

4.TUTO: Assemblage du prototype

Maintenant que nous disposons de tous les éléments dont nous avons besoin, nous pouvons brancher le tout. On branche à l’Arduino NANO sur le pin A0 et le pin GND aux pins de la masse et du bouton d’alimentation (respectivement le pin 30 et le pin 19) [10].

Grâce au programme écrit pour l’Arduino NANO, nous avons programmé des séquences d’allumage de la caméra. Nous avons maintenant une caméra qui prend trois photos toutes les 30 secondes dix fois de suite.



Montage de la Gopro relié à l'Arduino Nano

5. Alternative: ESP32-Cam

ESP32-Cam

Pour utiliser une carte SD avec un ESP32-CAM, il est nécessaire de la formater en FAT32 (Attention, même si la carte SD fait plus de 4Go, l’ESP32-CAM ne sait pas gérer plus avec le port intégré)

Programmation d’une séquence de test pour l’ESP32-CAM :

- Connexion de l’ESP32-CAM avec une cable usb-ttl à l’ordinateur :

o Broche rx sur u0t

o Broche tx sur u0r

- GND sur GND
- 5v sur 5v
- Relier les broches IO0 et GND

Dans le menu de sélection de la carte, choisir la carte “AI-Thinker ESP32-CAM” et le port COM sur lequel est connecté l’ESP32.

Récupérer le code disponible pour prendre une photo:

https://github.com/ursusnocte/ARE_Curious_Info-Moorev-Timer-Camera/blob/main/ESP32/ESP32_CAM_test_photo/ESP32_CAM_test_photo.ino

ou un équivalent de vidéo, en réalité plusieurs photos par secondes:

https://github.com/ursusnocte/ARE_Curious_Info-Moorev-Timer-Camera/blob/main/ESP32/ESP32_CAM_test_video/ESP32_CAM_test_video.ino

Après cela, il ne reste plus qu’à téléverser le code sur l’ESP32-CAM, si la connexion ne s’effectue pas directement, appuyer sur le bouton de reset du microcontrôleur.

Afin d’afficher les messages envoyées par la carte en débug, ouvrir la console série intégrée de l’IDE et choisir une vitesse de transfert de 115200.

Le code pour photo réalise une photo à chaque allumage ou redémarrage de la carte et celui-ci pour la “vidéo” prend 10 images par secondes dans la configuration actuelle jusqu’au débranchement de la carte.

Pour comprendre plus en détail le fonctionnement des scripts, je vous conseille l’article sur l’ESP32-CAM en référence [11].

6. Références:

Référence de l’ESP32:

[0] https://www.adafruit.com/product/3405

Téléchargement de l’IDE:

[1] https://www.arduino.cc/en/software

Référence sur les modes de l’ESP32 pour le code:

[2] https://letmeknow.fr/fr/blog/142-tutoriel-les-sleep-modes-de-lesp32

Lien vers notre GitHub:

[3] https://github.com/ursusnocte/ARE_Curious_Info-Moorev-Timer-Camera

Lien pour alimentation de l’Arduino:

[4]https://idehack.com/alimenter-larduino-nano/

Lien pour branchement ESP32:

[5] https://learn.adafruit.com/adafruit-huzzah32-esp32-feather/power-management

Lien vers le modèle du connecteur Herobus:

[6] https://www.digikey.fr/en/products/detail/jae-electronics/DD1P030MA1/1283154

Lien vers les descriptions des pins du connecteur arrière de la caméra:

[7] https://mewpro.cc/en/2014/10/14/herobus-pinout-of-gopro-hero-3-black/

Référence autoexec scripts:

[8] https://github.com/KonradIT/autoexechack

Lien vers les informations concernant le connecteur herobus:

[10] https://github.com/KonradIT/gopro-herobus-sdk/blob/master/diyprojects.md

Lien vers le site de Camdo sur la batterie des caméras:

[9] https://cam-do.com/pages/power-consumption-by-gopro-camera-model

Lien vers l’article sur l’ESP32-Cam :

[11] https://randomnerdtutorials.com/esp32-cam-take-photo-save-microsd-card/

Revision #17
Created 30 March 2023 07:50:11 by Chu Amelie
Updated 19 July 2023 10:33:55 by Chu Amelie