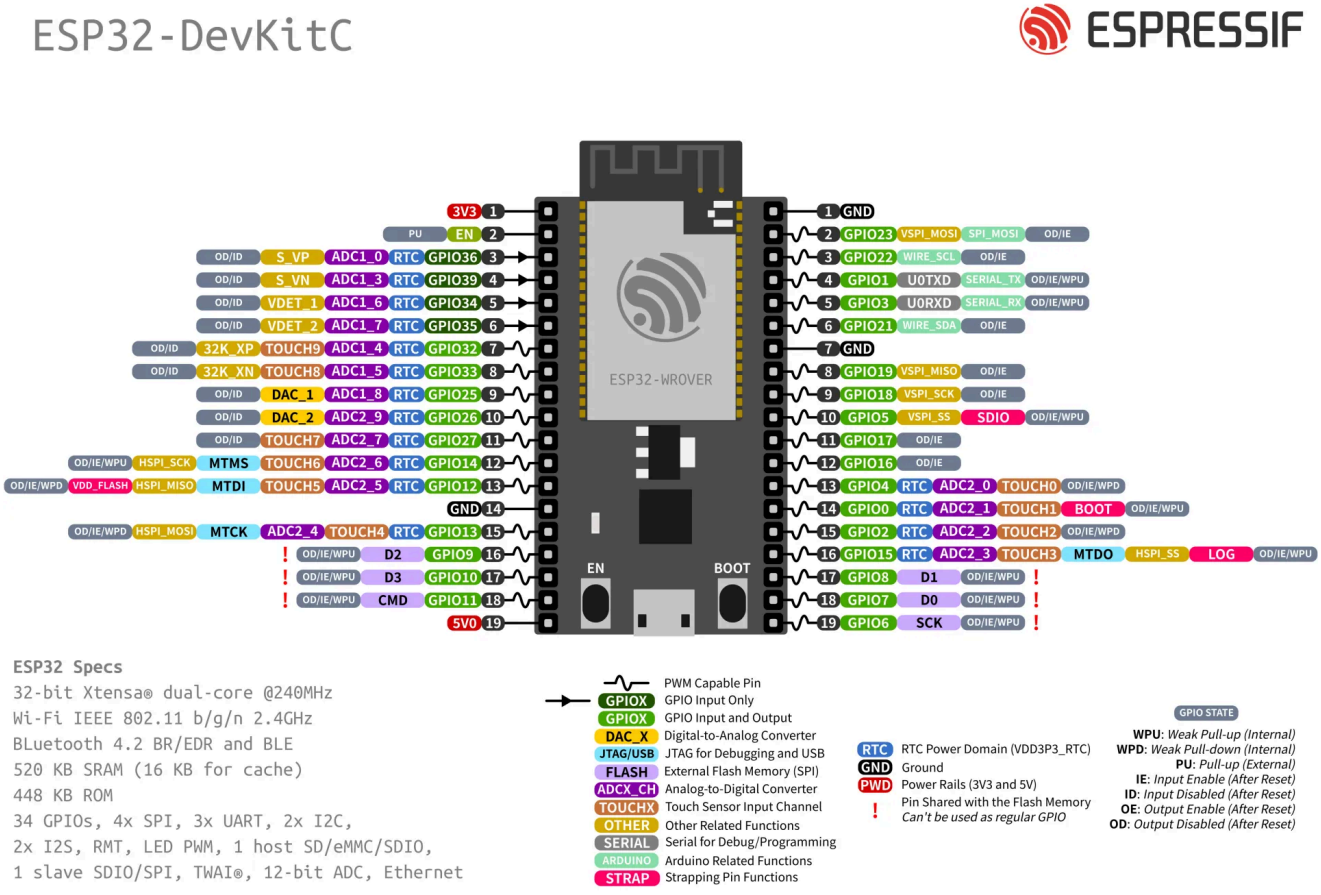
LILIT-2026_ESP32

alain.empain@uliege.be

Généralité

Constructeur : Espressif (cf ESP32) : https://www.espressif.com/

Microcontroleur procurant un grand nombre de pattes multiplexées, programmables individuellement en input/outpout/ADC/DAC/touch..., bus I2C, I2S, serial (hardware sur pattes dédiées ou software)...
(avec certaines restrictions, pas toutes pour toutes les fonctions)



Alimentation en 5VDC, par câble USB ou directement sur les pattes
Fonctionnement en 3.3VDC

Très faible consommation en mode DeepSleep

Au départ : ESP8266 juste comme périphérique pour fournir la WiFi (commande à ditance, ampoules). La puissance embarquée a été remarquée et diverses applications autonomes ont été développées.

Avec la génération suivante ESP32, la puissance fournie permet bien d'autre choses !

Comparaison

		RAM kB	MHz	CPU	Storage	Network
Commodore-64	27000 BEF ~ **1673 €**	64	2	8 bits	170 kB (floppy)	-
PC-XT	150000 BEF ~ **8365 €**	128-512	4.77	8 bits	floppy HD 10 MB	-
ESP-32	5 à 15 €	160-8000 (typ 520)	160-400	32 bits	Flash 4 GB (opt: microSD)	WiFi (5/6) ESPNow BlueTooth *classic* BLE (opt:LoRa)

A ce prix là et avec toutes ces caractéristiques, cela mérite un maximum d'attention

Il est temps de reconsidérer avec respect l'efficience des anciens programmes :

Linux version TRINUX tenait sur un floppy 3.5", y compris ethernet, plein d'outils standard, avec deux floppies supplémentaires pour plus applications.

Un floppy 3.5" de démo d'un UNIX industriel (=QNX=, 1998) suffisait pour contenir les outils standard, plus X-Window minimaliste, ethernet, et même... un browser décent.

Diversité

https://www.espboards.dev/blog/esp32-soc-options/

Quick Navigation: Choose Your ESP32

Modèle	CPU	SRAM/Flash	Wi-Fi	Bluetooth	802.15.4 (Zigbee/Thread)	Comme
ESP32	Xtensa LX6 dual 240 MHz	520 KB SRAM, ext. Flash	802.11b/g/n	BT4.2 + BLE	–	Dual-co
ESP32-S2	Xtensa LX7 single 240 MHz	320 KB SRAM	802.11b/g/n	– (aucun)	–	Wi-Fi or
ESP32-S3	Xtensa LX7 dual 240 MHz	512 KB SRAM (+ PSRAM)	802.11b/g/n	BLE 5.0	–	IA/visior vs-esp3 best/#:~

Modèle	CPU	SRAM/Flash	Wi-Fi	Bluetooth	802.15.4 (Zigbee/Thread)	Comme
ESP32-C3	RISC-V single 160 MHz	400 KB SRAM	802.11b/g/n	BLE 5.0	–	Petit, sé
ESP32-C6	RISC-V dual (160+20 MHz)	512 KB SRAM	Wi-Fi 6 (2.4GHz)	BLE 5.x	IEEE 802.15.4	Matter/T energy6 c6#:~:te
ESP32-C61	RISC-V single 160 MHz	320 KB SRAM	Wi-Fi 6 (2.4GHz)	BLE 5.0	–	Variante
ESP32-C5	RISC-V single 240 MHz (+LP 40)	384 KB SRAM	Wi-Fi 6 (2.4/5GHz)	BLE 5.0	IEEE 802.15.4	Dual-ba c5#:~:te
ESP32-H2	RISC-V single 96 MHz	320 KB SRAM	–	BLE 5.0	IEEE 802.15.4	Zigbee/T h2#:~:te
ESP32-P4	RISC-V dual 400 MHz + LP 40 MHz	768 KB SRAM	–	–	–	HMI/AI,

Processeur

- Famille **ARM** classique (comme dans tous les smartphones, ou les nouveaux MACs)
- Famille **RISC-V,** révolutionnaire (micro-code open-source, pas de royalties à payer). Voir par ex la série XIAO-C3 et XIAO-C6

  ==Cet ecosystème homogène permet de n'avoir qu'une chaîne de développement pour produire une multitude de chips spécialisés à intégrer dans un SoC, à la différence des systèmes classiques.

En principe **deux CPUs**, permettant de gérer plusieurs jobs à la fois, même en real-time,

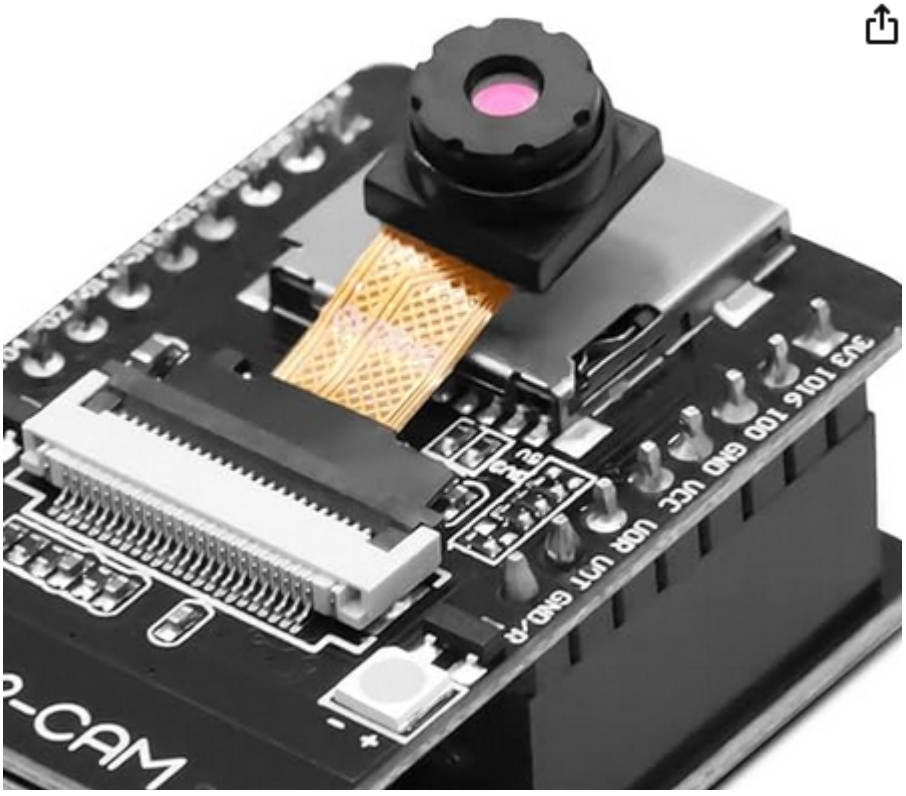
Plus un CPU à très basse consommation pour le maintien de l'activité de base (timer...) pendant les DEEPSLEEPs, permettant d'éveiller le système de temps en temps et prolonger la vie des batteries.

Taille finale (chip ESP-32 + ses périphériques de base)

Famille XIAO de Seeed :

VARIANTS

Existe avec camera, LoRa, micro-écran... on-board



Une version remarquable :

CYD (CheapYellowDisplay####

< 20€), ESP32 classique + un écran tactile de 2.8", mais peu de GPIOs accessibles.
Voir plus loin un exemple d'application avec la librairie graphique LVGL

RAM/ROM

Variable suivant les options de base, mais toujours en quantité confortable

FOURNISSEURS

Nombreux, il est possible d'éviter de commander hors EU, par ex commande directe en Allemagne :

Seeed (CN, DE) : https://www.seeedstudio.com/eu_warehouse
AZ-Delivery (DE) : https://www.az-delivery.de/fr/pages/search-results-page?collection=esp32
Tutoriaux très intéressants

Communication

Wifi,

- classique, version 6 supportée par le petit dernier ESP32
- protocole ESP-Now : surcouche avec très faible latence, pour message courts et portée allongée
  BlueTooth
  LoRa (on-board sur certains systèmes, ou en périphérique)

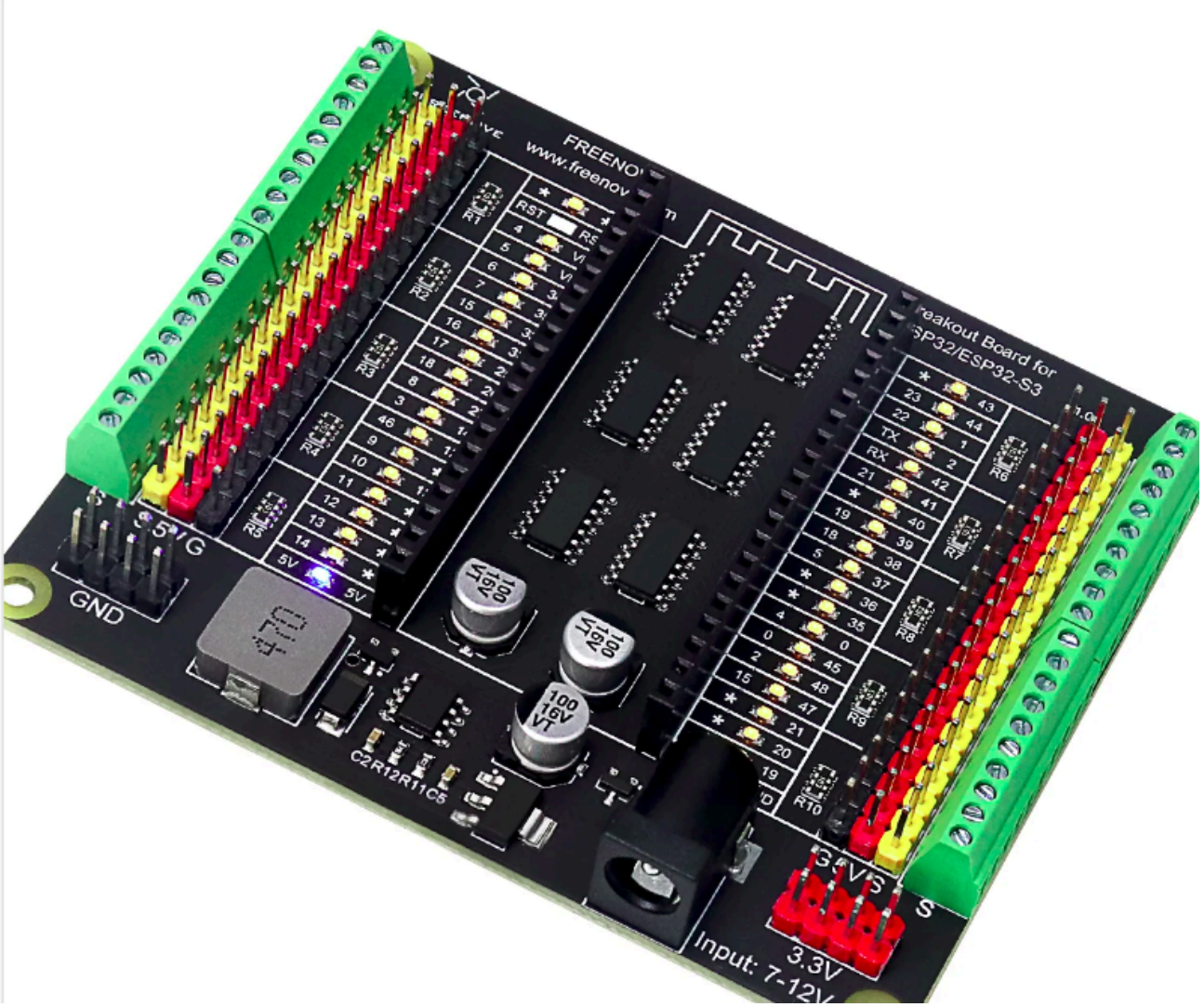
Serial
USB/serial : connection de base, de PC à l'interface de conversion USB/serial on-board

BUS

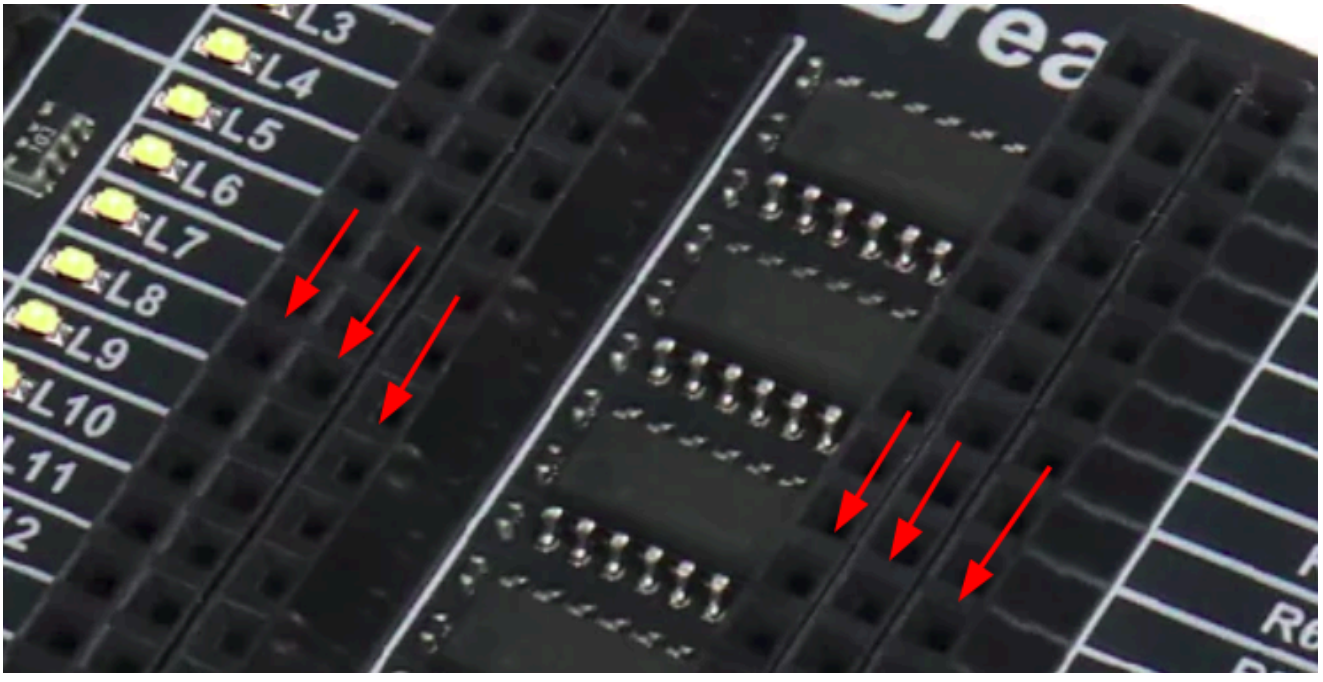
supporte divers bus classiques : I2C, I2S, OneWire...

Supports d'extension

Génériques, avec leds pour signaler l'état ON/OFF de chaque patte (< 20€)
https://freenove.shop/products/fnk0091?v_id=variant_01KPCTTWPS052KEBCKKADVX34G



Universal : support divers écartements entre les pattes (deux fois trois rangées de connecteurs)

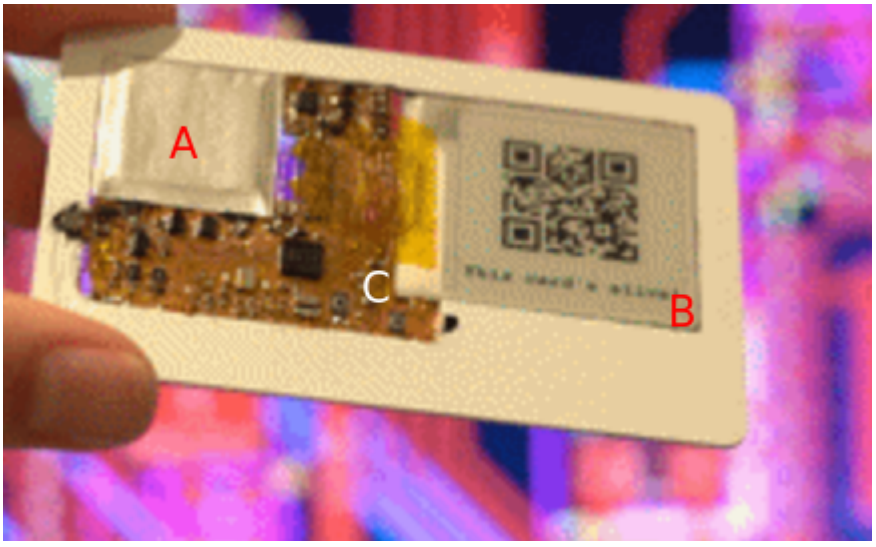


Ex pour XIAO (dans le cercle), avec écran

Dernière nouvelle, merci RR de m'avoir transmis l'information d'un membre de LgHS !
https://github.com/krauseler/muxcard

Une carte de visite au format d'une carte banquaire, 1 mm d'épaisseur !

- A: batterie ultra fine
- B: écran e-Paper
- C: ESP-C3 (processeur RISC-V)



Langages

1- La base, un OS 'real time' : freeRTOS

A Real-Time Operating System (RTOS) is a type of computer operating system designed to be small and deterministic. RTOSes are commonly used in embedded systems such as medical devices and automotive ECUs that need to react to external events within strict time constraints.

2- Langages

Langages compilés

- Approche de référence : les librairies sont en principe développées à l'origine en C
  Librairie graphique très complète et légère : LVGL voir dans micropython
- Vu aussi une référence
  - au langage **RUST**,
  - et aussi à **GO** (tinyGo ?)

Interprétés

- **FORTH** : compilation incrémentielle, donc conserve la rapidité d'un langage compilé, avec la facilité d'un langage interprété.
  Support très actif pour ESPForth, ''
- **micropython** : un variant de Python 3 en pratique très semblable à l'original

FORTH

Livres gratuits en anglais et français (l'auteur Marc Petremann est francophone):

Option très intéressante, très rapide, supporte de nombreux aspects de ESP32, dont LoRa, capture d'image (CAM-ESP) et ESP-NOW, mais toutes les librairies pour périphériques (sondes...) ne sont pas encore développées.

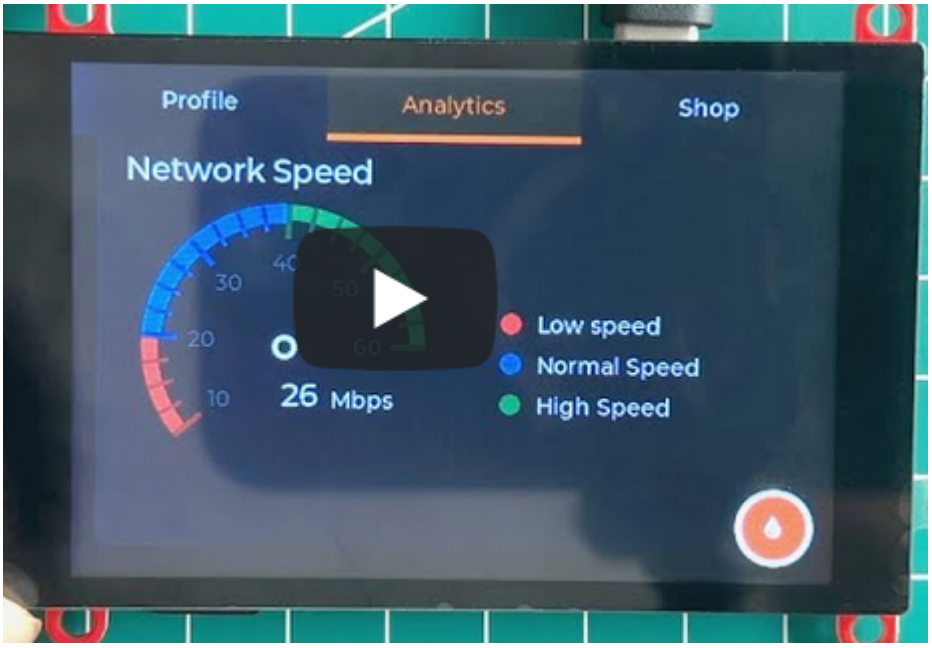
Source, avec dans le Readme la présentation globale de ce qui est disponible
https://github.com/MPETREMANN11/ESP32forth/

MicroPython

Approche très intéressante pour développer de façon interactive.
Moins rapide que FORTH, mais en général tous les pilotes (bus, sondes...) ont été portés en micropython, ce qui est très confortable.

Librairie graphique LVGL
https://lvgl.io/docs/open/9.3/details/integration/bindings/micropython

Un apport important, cette librairie 'légère' mais puissante est utilisée par ex dans les montres connectées.
Elle procure un ensemble important de widgets (librairie en C,, donc compilée et rapide),



Pas besoin d'installer LVGL pour essayer, il y a un émulateur sur le web :
https://sim.lvgl.io/v9.0/micropython/ports/webassembly/index.html

---

Environnement intégré : ESP-home

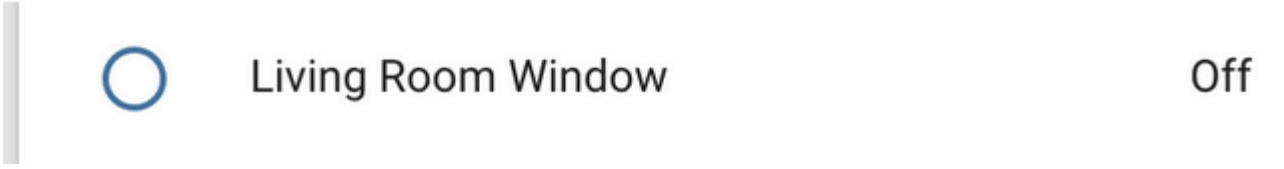
Smart Home made easy : https://esphome.io/
On flashe une image générique compilée pour la cible (ESP32 xx), et on définit la configuration et la fonctionnalité dans un fichier de définition
(format YAML (nom récursif : "YAML Ain't Markup Language")

"Turn your ESP32, ESP8266, BK72xx, RP2040, and other supported boards into powerful smart home devices with simple YAML configuration."

Next, let's add a binary sensor which will monitor a GPIO pin to determine and report its state.

```
binary_sensor:
  - platform: gpio
    name: "Living Room Window"
    pin:
      number: GPIO0
      inverted: true
      mode:
        input: true
        pullup: true
```

In Home Assistant, the example code above will look like this:



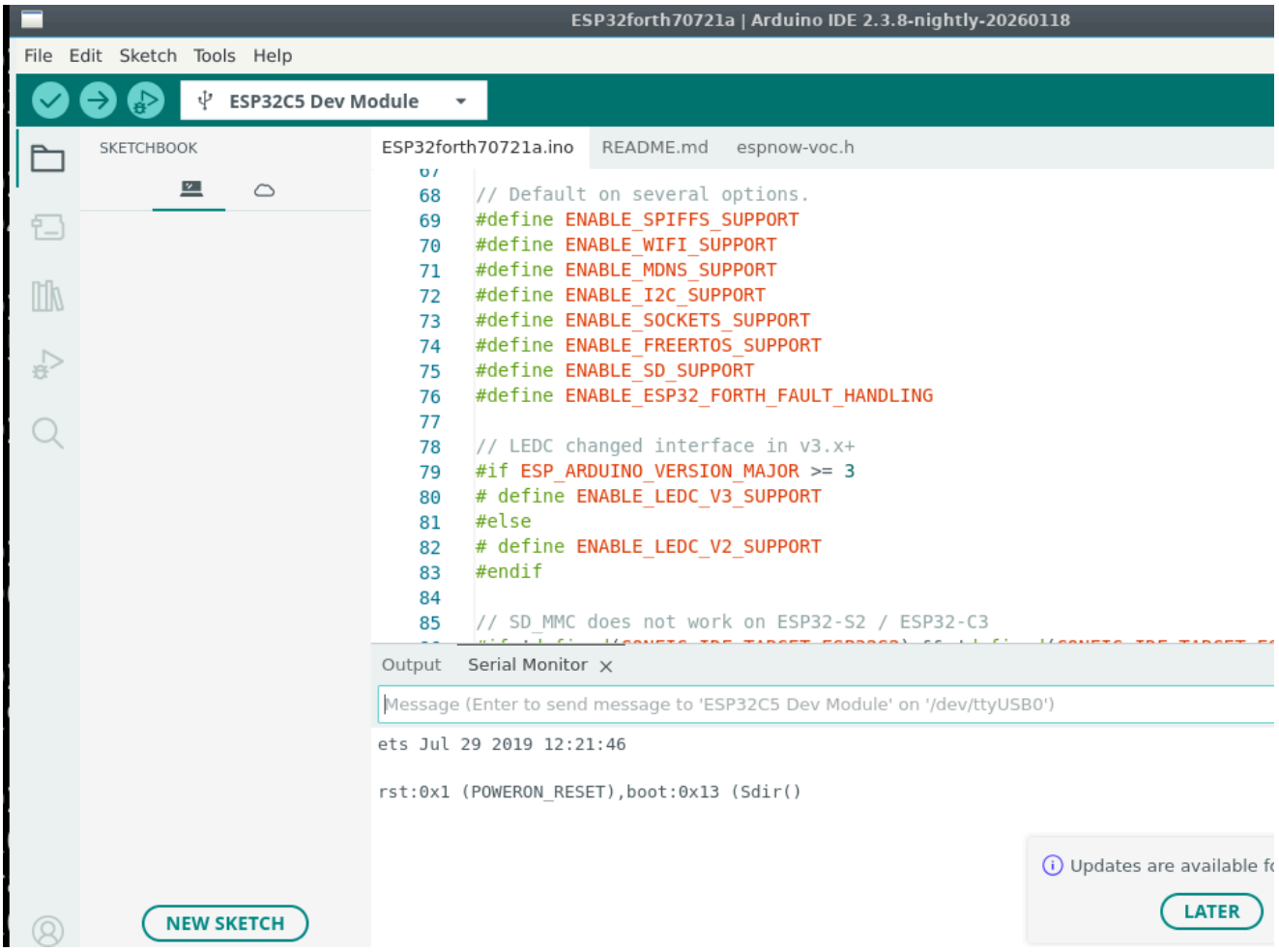
Environnement de développement

Graphiques

ESP-IDE, la référence du constructeur ESPRESSIF
https://www.espide.eu/en/

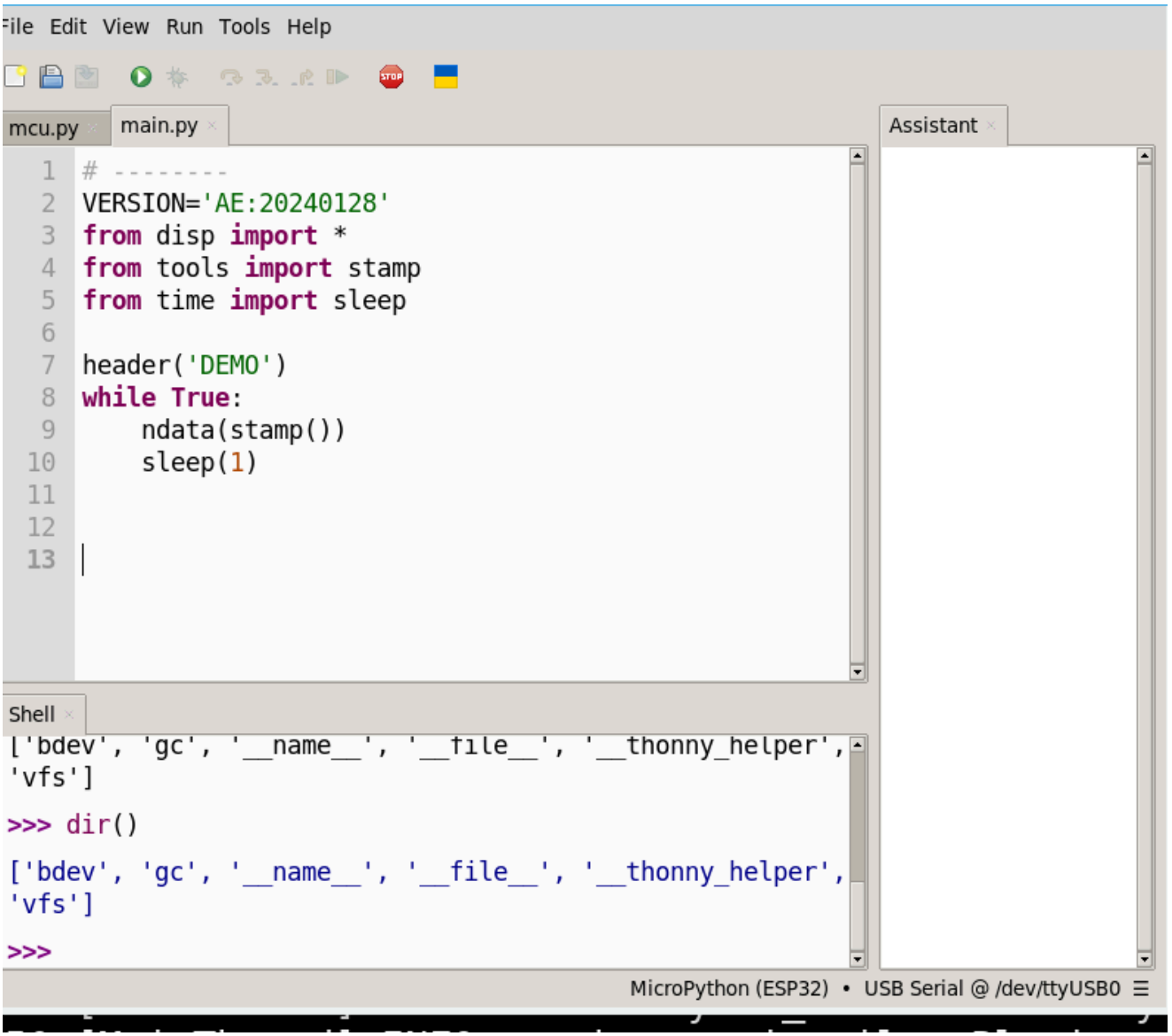
Arduino-IDE, le classique pour Arduino et autres
https://www.raspberrypi-france.fr/tutoriel-arduino-ide-guide-complet-en-francais/

je l'utilise en particulier pour compiler le FORTH et le flasher sur le ESP32 cible



THONNY (pour python, pas pour compiler et flasher)

thonny.org

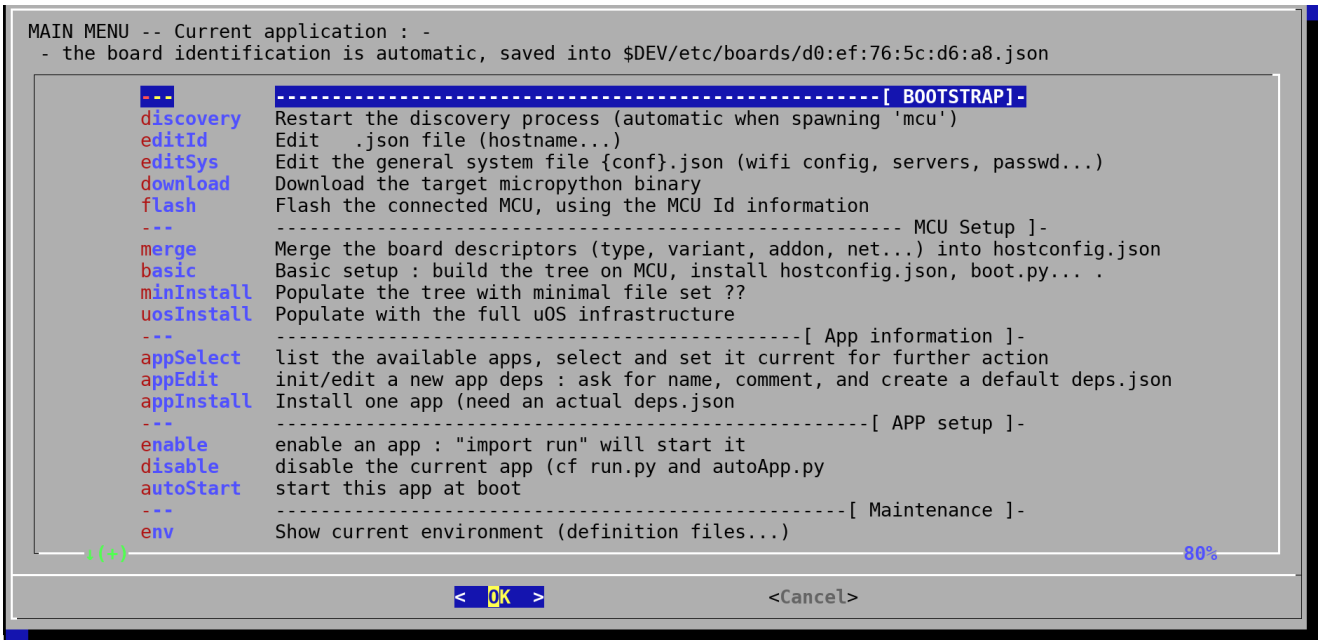


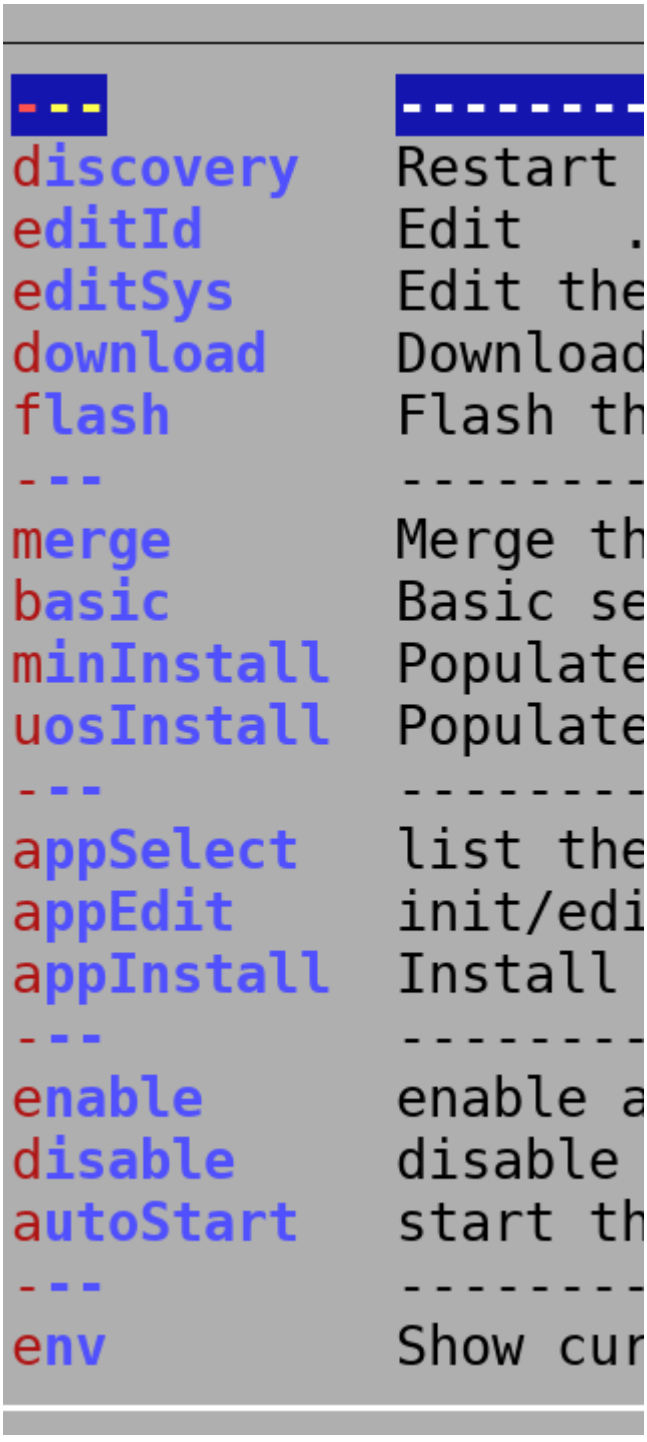
Gestion en ligne de commande

Utilitaires de base (espressif):

- **esptool** : pour flasher et agir en bas niveau sur le ESP32 https://docs.espressif.com/projects/esptool/en/latest/esp32/
- **mpremote** : pour les interactions de haut niveau (copie de fichiers, passage en mode moniteur...) https://docs.micropython.org/en/latest/reference/mpremote.html
- **mpr** : simple variant de mpremote, avec un interface cadré sur les conventions classiques sous shell (bash) https://pypi.org/project/mpr/

'MCU', mon interface avec menu pour gérer les opérations de base


MAIN MENU -- Current application : -
- the board identification is automatic, saved into $DEV/etc/boards/d0:ef:76:5c:d6:a8.json
--- ----------[BOOTSTRAP]-
discovery Restart the discovery process (automatic when spawning 'mcu')
editId Edit .json file (hostname...)
editSys Edit the general system file {conf}.json (wifi config, servers, passwd...)
download Download the target micropython binary
flash Flash the connected MCU, using the MCU Id information
--- ---------- MCU Setup]-
merge Merge the board descriptors (type, variant, addon, net...) into hostconfig.json
basic Basic setup : build the tree on MCU, install hostconfig.json, boot.py... .
minInstall Populate the tree with minimal file set ??
uosInstall Populate with the full uOS infrastructure
--- ----------[App information]-
appSelect list the available apps, select and set it current for further action
appEdit init/edit a new app deps : ask for name, comment, and create a default deps.json
appInstall Install one app (need an actual deps.json
--- ----------[APP setup]-
enable enable an app : "import run" will start it
disable disable the current app (cf run.py and autoApp.py
autoStart start this app at boot
--- ----------[Maintenance]-
env Show current environment (definition files...)
80%
< OK >
<Cancel>


discovery
editId
editSys
download
flash
merge
basic
minInstall
uosInstall
appSelect
appEdit
appInstall
enable
disable
autoStart
env

Exemple de cycle de développement avec les outils de base standard (CLI)

PC: edition du script.py (ex: 'vi script.py')
PC: copie sur le file system de l'ESP, destination /app) : ' mpr put script.py /app '
PC: passage en mode interactif sur l'ESP : ' mpr remote' ou plus court 'mpr r'
On est alors sur l'ESP !
ESP>>> *dir()* # liste ce qui est connu dans la liste locale
ESP>>> *import math*
ESP>>> *dir(math)* # liste tout ce qu'il connait du module 'math'
ESP>>> *math.sqrt(10.23)*
3.1984371183438953

REM: # variant :
ESP>>> *from math import sqrt*
on évite ainsi de devoir utiliser math.sqrt()
on tape simplement :
ESP>>> sqrt(10.23) # ok, il trouve bienn sqrt(), vérifier avec dir()

On charge/exécute script.py, tout comme on avait chargé le module math.py

soit
ESP>>> import app.script # REM: un point pour le path, pas de / ni de .py !!
soit
ESP>>> from app.script import * # REM: astérisque = 'all'

```
On teste,
on revient au PC avec CTRL-X

___
On réédite sur PC,
on copie sur ESP avec 'mpr put' et
on reteste en revenant sur l'ESP avec 'mpr r'
___
```

REM: il est possible d'éditer DIRECTEMENT sur ESP, mais alors on n'a pas de trace sur le PC ! 'mpr edit /app/script.py'

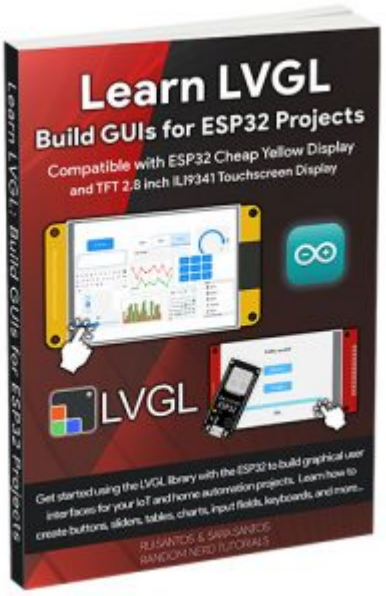
Resource : Random Nerd Tutorials

https://randomnerdtutorials.com/projects-esp32/

Ruis est très sympa, qui répond quand on lui écrit ;-)
Free ebooks : https://randomnerdtutorials.com/download/

Shop Random Nerd Tutorials Courses

Showing all 12 results

- [

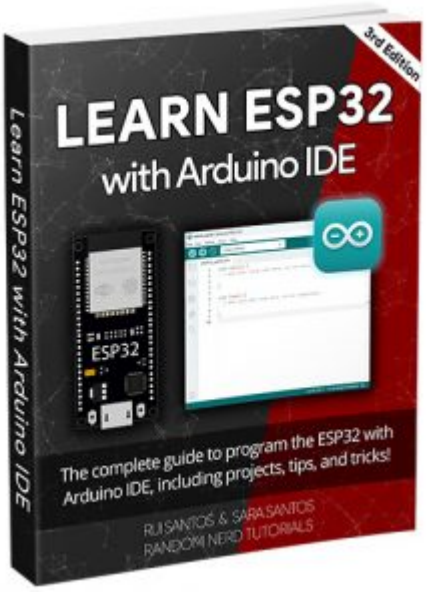
Learn LVGL: Build GUIs for ESP32 Projects (eBook)

$37.00](https://rntlab.com/product/learn-lvgl-build-guis-for-esp32-projects-ebook/) View Course »

- [

Learn Raspberry Pi Pico/Pico W with MicroPython (eBook)

$27.00](https://rntlab.com/product/learn-raspberry-pi-pico-w-with-micropython-ebook/) View Course »



- [

Learn ESP32 with Arduino IDE eBook | 3rd Edition

$27.00](https://rntlab.com/product/learn-esp32-with-arduino-ide/) View Course »

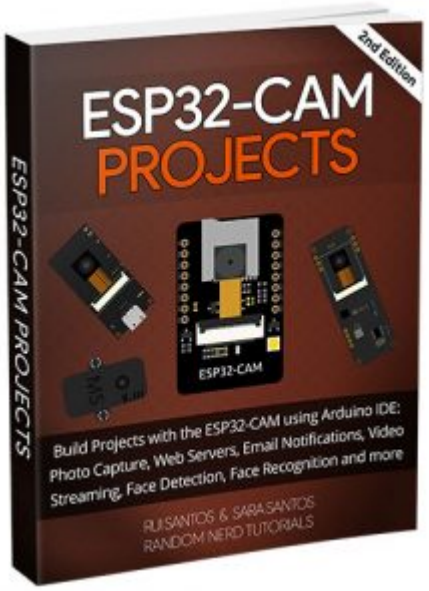
- [

SMART HOME with Raspberry Pi, ESP32, and ESP8266 eBook

$27.00](https://rntlab.com/product/smart-home-with-raspberry-pi-esp32-and-esp8266-ebook/) View Course »

- [
  ## Home Automation using ESP8266 | 4th Edition (eBook + mini video course)
  $18.00](https://rntlab.com/product/home-automation-using-esp8266/) View Course »

- [
  ## Build Web Servers with ESP32 and ESP8266 eBook | 3rd Edition
  $27.00](https://rntlab.com/product/build-web-servers-esp32-esp8266-ebook/) View Course »

- [

Build ESP32-CAM Projects using Arduino IDE eBook | 2nd Edition

$29.00](https://rntlab.com/product/esp32-cam-projects-ebook/) View Course »

- [

Firebase Web App with ESP32 and ESP8266 eBook

$29.00](https://rntlab.com/product/firebase-web-app-with-esp32-and-esp8266-ebook/)
View Course »

- [

  MicroPython Programming with ESP32 and ESP8266 eBook | 2nd Edition
  $24.00](https://rntlab.com/product/micropython-programming-with-esp32-and-esp8266/)
  View Course »

- [

OBJ

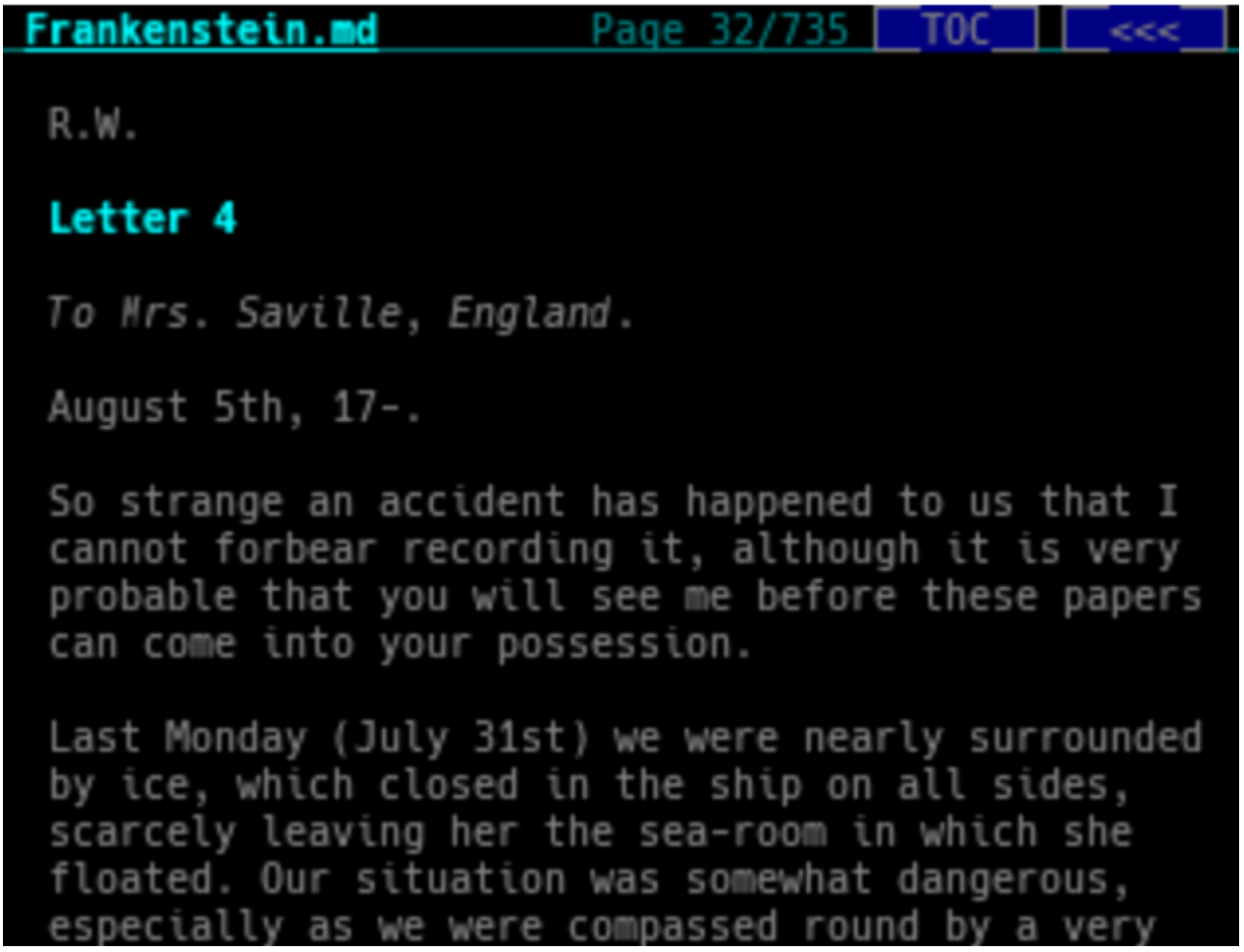
Dépasser les limites !

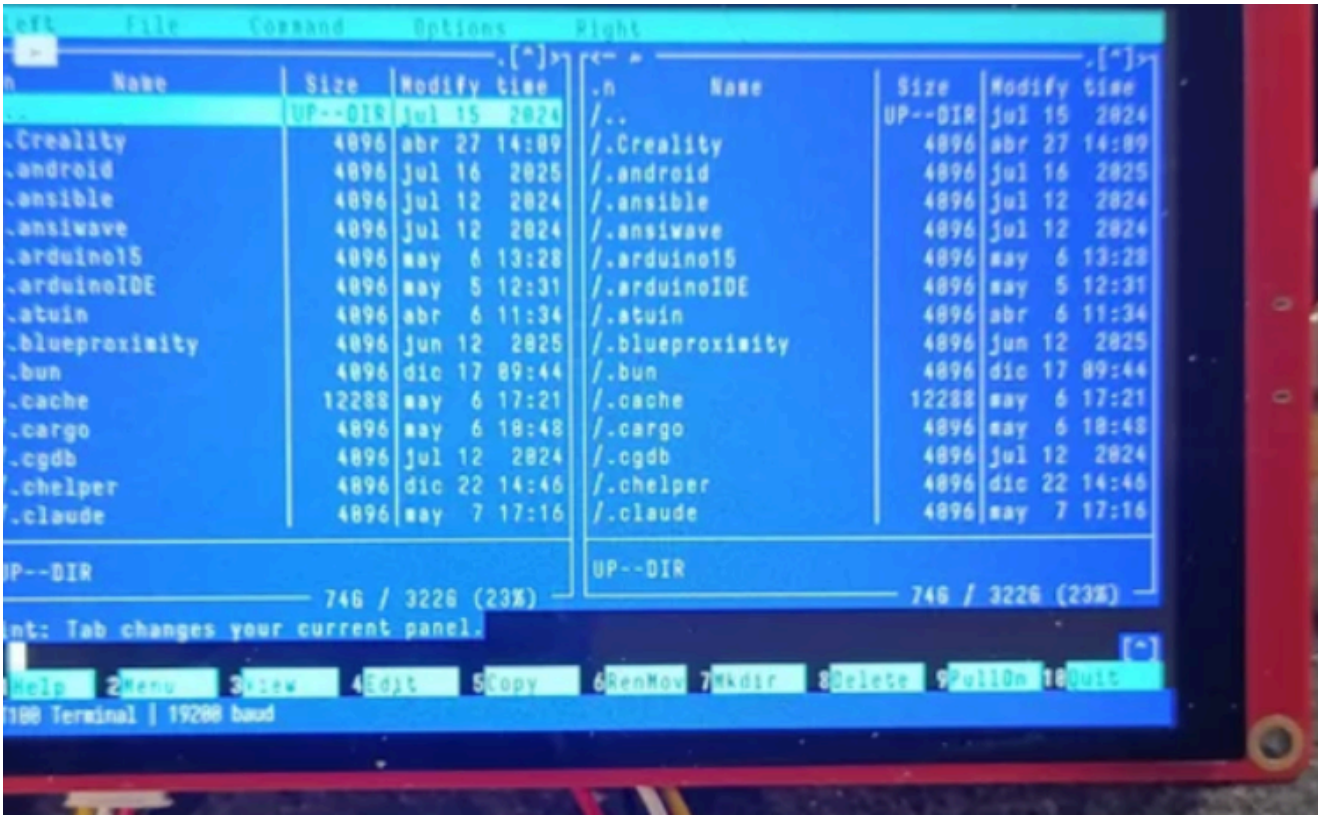
ESP-OSito : créer un environnement semblable à celui des machines de fin des années '80' (single user/single task) : compiler chaque tâche, les stocker dans l'espace de fichiers (sur flash) et les excécuter à la demande comme sur un PC sous DOS...

REM: les outils de GNU-Linux sont open-source, il y a une mine à exploiter relativement facilement !

https://hackaday.com/2026/05/29/esp-osito-eschews-retrocomputing-for-modern-code-on-modern-equivalent-hardware/

Activer une lectrice de bouquins (au format 'md = MarkDown), ou midnight commander etc. voir la liste actuelle sur le site de l'auteur https://esposito.ralsina.me/





Exemples d'applications personnelles

A.Empain 2026-06-01

Surveillance de chaudière

Surveillance sans toucher aux câblages : phototransistors glissés dans des trous d'une barre de plexi positionnée devant les LEDs du tableau de la chaudière + sondes de température (ancien projet Arduino en voie de migration)

Gestion de deux caissons pour semis et bouturages

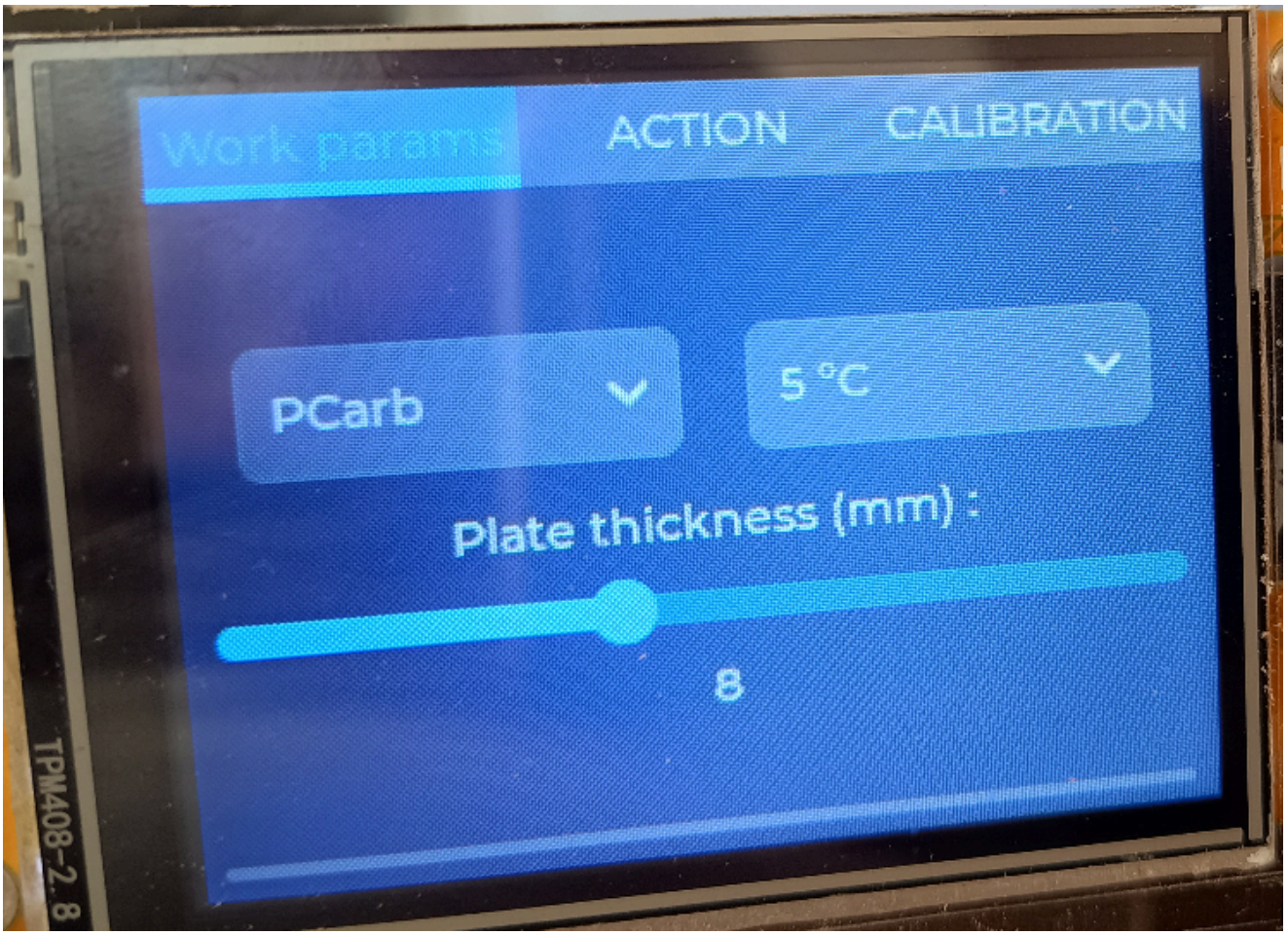
- deux enceintes indépendantes, total ~ 1/2 m³
- contrôle de la température intérieure, consigne de jour (max) et de nuit (min)
- contrôle de l'éclairage
- mesure de l'humidité relative
- mesure de la température extérieure
- log des mesures -> ESP-NOW vers un centralisateur dans la maison (CheapYellowDisplay)

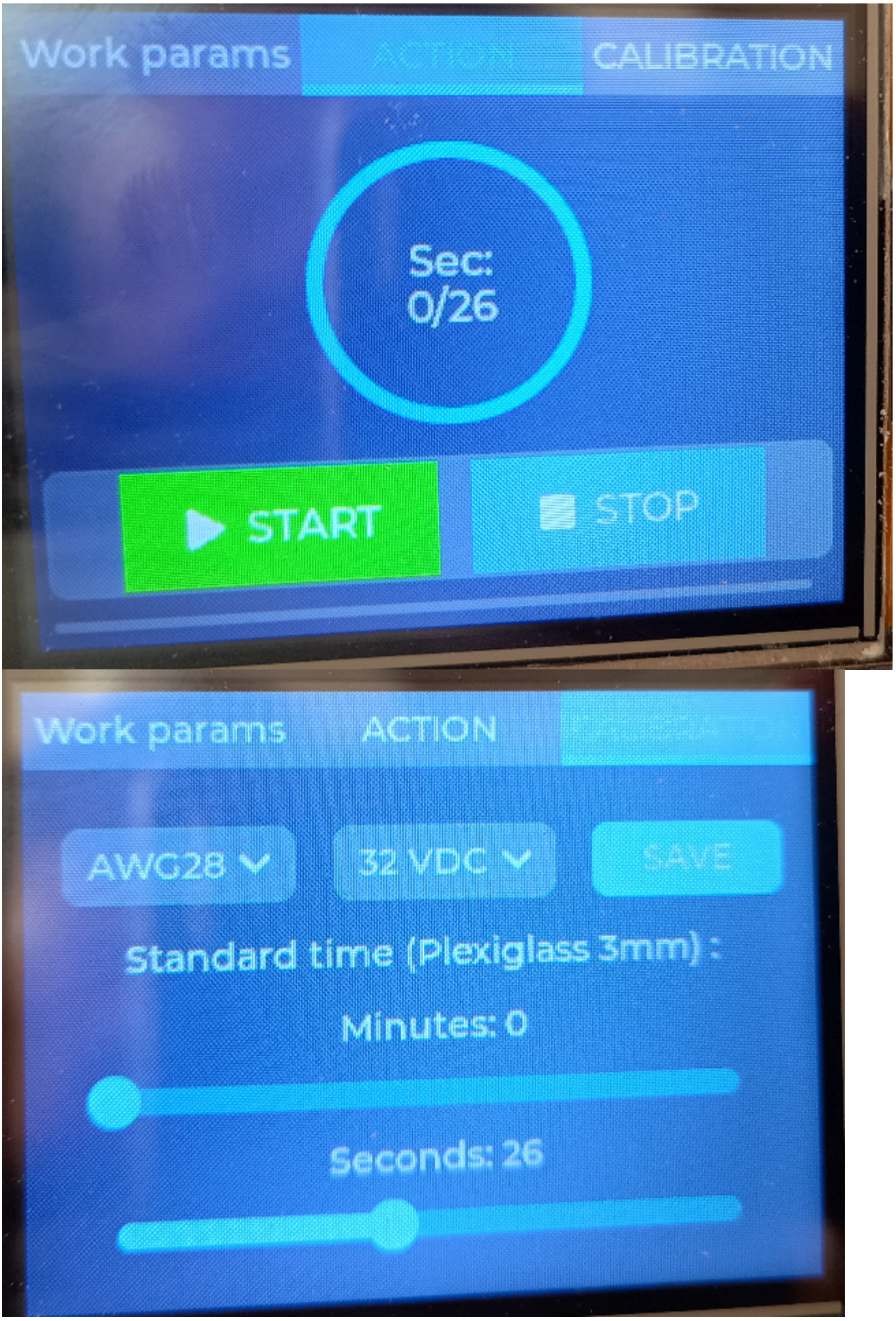
CheapYellowDisplay pour piloter une plieuse de plastic

Trois frames de menu

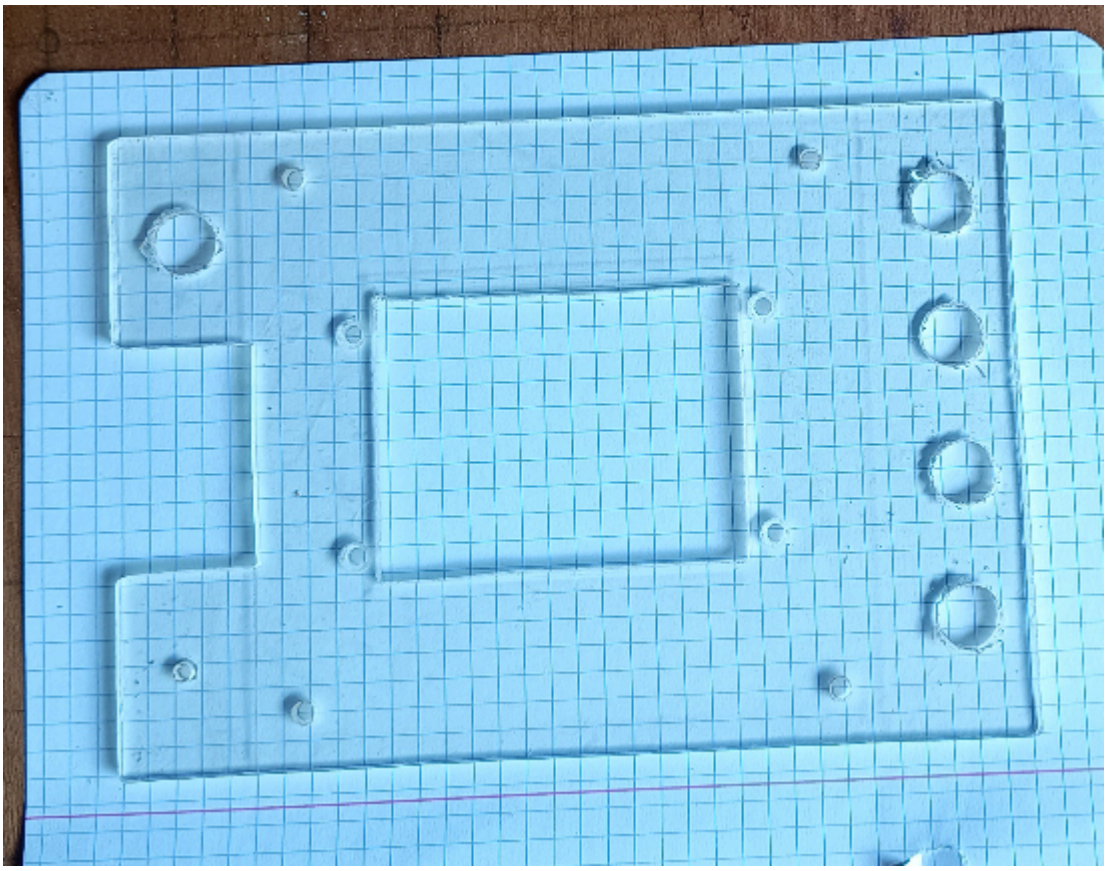
```
Setup de travail :
    type de plastic (plexi, PVC, polyéthylene...)
    épaisseur,
    température ambiante pour correction de temps
Action :
    start (start/stop automatiques)
    stop (pour interruption en cours de cycle)
Calibrages du système :
    épaisseur du fil NiChrome
    voltage de la source
```

```
    temps standard pour plier 3mm de plexi (min et secondes)
     [SAVE] sauve le tout dans un fichier au format JSON sur le stockage
permanent (flash disk), rechargé automatiquement au démarrage.
```

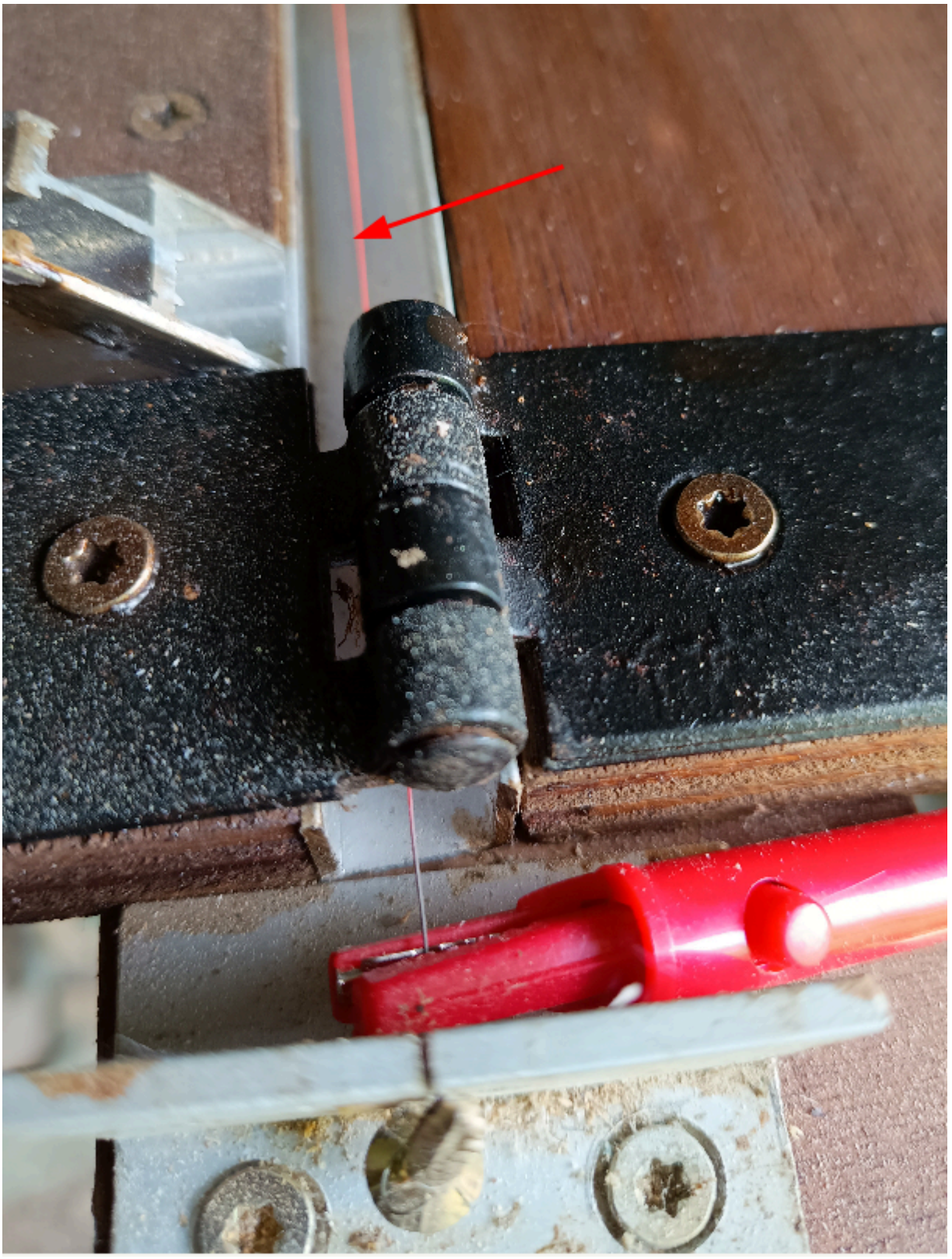




Exemple de réalisation



Plieuse : deux plaques de bois articulées avec charnière, une cornière en U avec un fil en nickel-chrome tendu dans la gorge du U



Je place la plaque de plexi le long du guide, en alignant le trait à plier sur le fil

J'attends le temps nécessaire (2min15 pour 3mm de plexi), et je plie grâce aux charnières, et tenant bien en place le côté fixe du plexi.

Comme la partie chauffée est très bien localisée, le pli est impeccablement aligné !

Bien maintenir l'angle voulu pendant le refroidissement (~ une minute) : un simple bloc de bois fait l'affaire.

Et voilà ! Il ne reste plus qu'à y visser le CheapYellowDisplay de la démo.