Une image contenant texte, équipement électronique, capture d’écran

Description générée automatiquement

Lecteur RFID UHF V3 éducation

rapport de modifications

Cruz Pastor, Dietrich, Carballo | ISE-ISC | 14.3.2022

Table des matières

[Introduction 2](#_Toc101713684)

[Recherche de composants et du module WIFI/Bluetooth 2](#_Toc101713685)

[Modification du schéma électrique 2](#_Toc101713686)

[Réalisations empreintes 4](#_Toc101713687)

[Emplacement et routage PCB 6](#_Toc101713688)

[Réalisation et test de fonctionnement 8](#_Toc101713689)

[Conclusion 8](#_Toc101713690)

[Annexe 9](#_Toc101713691)

[GUIDE ESP32-C3-WROOM-02 9](#_Toc101713692)

# Introduction

Dans ce travail, il nous a été demandé de modifier la carte réalisée par Gaëtan Passeri pour son travail de bachelor, qui est un lecteur RFID UHF.

Les modifications à réaliser sont de rajouter une interface WIFI/Bluetooth, supprimer les points de test superflu et de déplacer l’écran vers le centre de la carte. Dans un second temps nous allons identifier les composants qui ne sont plus disponibles et les replacer par des équivalents. Le schéma et le PCB devront être adaptés avec ces nouveaux composants et dans un souci d’écologie et de réparabilité nous devons choisir des composants conventionnels et simples à remplacer.

Finalement nous allons envoyer notre carte en production, la monter et la tester.

# Recherche de composants et du module WIFI/Bluetooth

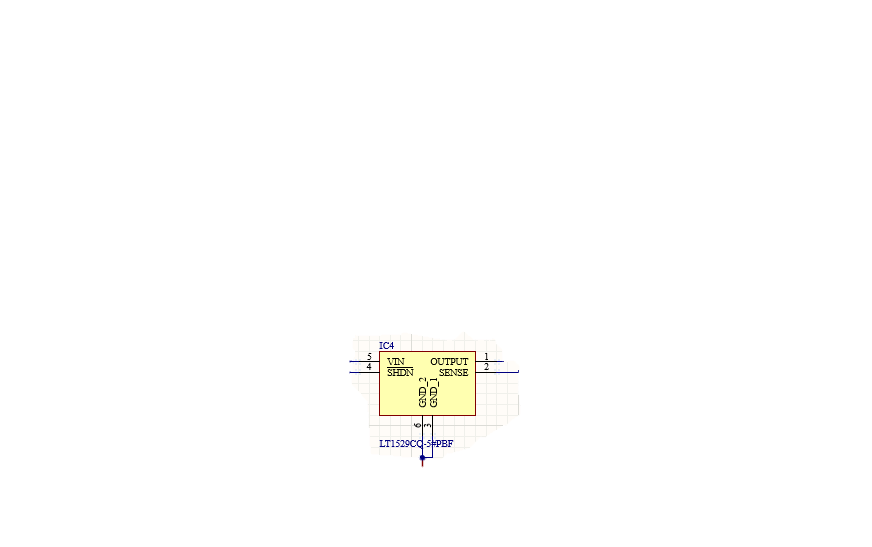
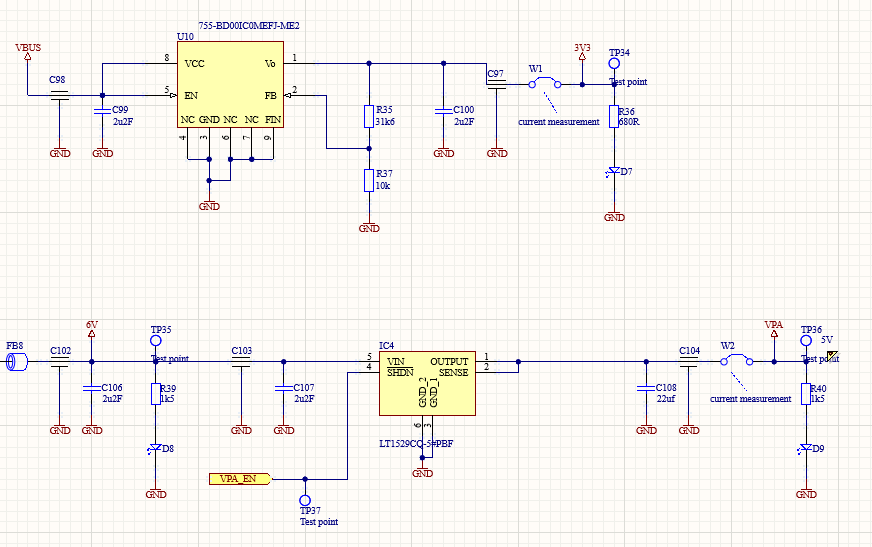
Voici les composants que nous avons dû modifier :

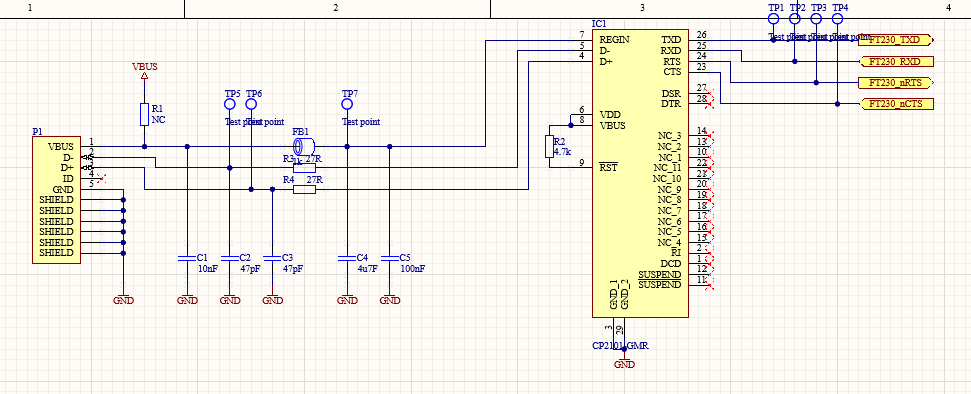
* Des condensateurs 5,6[pF]
* Une diode Schottky
* Des billes de ferrite de 1 [kΩ]
* Des régulateurs de tension de 3,3 et 5 [V]
* Une interface pour la communication UART/USB
* Un condensateur programmable par SPI
* Un quartz de 32,768 [KHz]

Il faut aussi rechercher un module pour le WIFI/Bluetooth. Pour cela nous avons sélectionné le ESP32-C3-WROOM-02-N4 avec les composants nécessaires pour son bon fonctionnement.

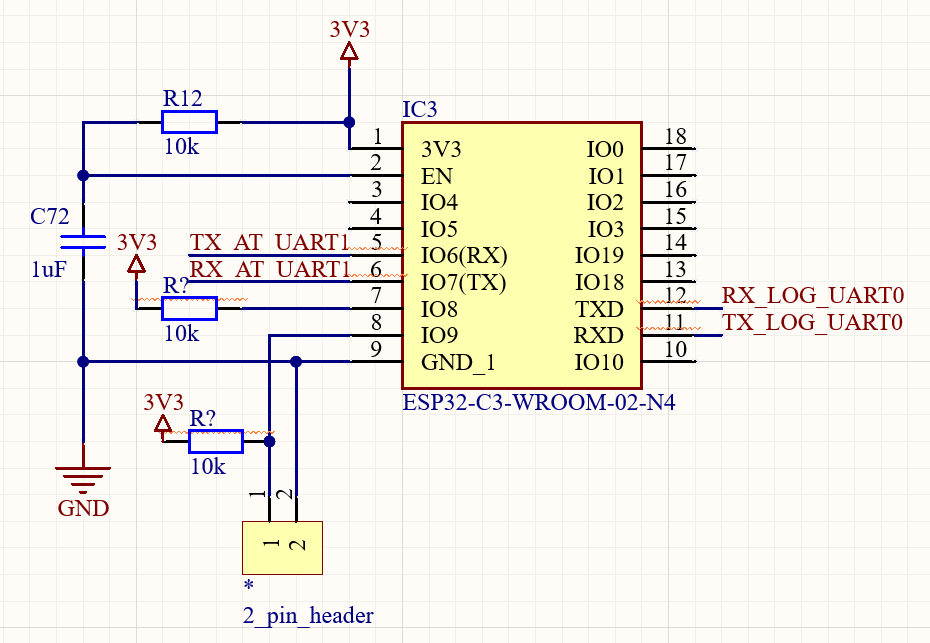
# Modification du schéma électrique

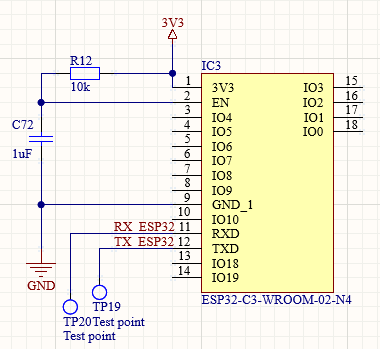
Certains composants avaient les mêmes boitiers que leurs originaux, néanmoins les composants que nous avons dû remplacer sont les régulateurs de tension pour avoir un ampérage plus grand et le pont UART-USB.

  
*Régulateurs de tension 3.3 (à gauche) et 5 volts.*

  
*Pont USB-UART.*

Nous avons supprimé un grand nombre de points de test en gardant les principaux. Ensuite nous avons supprimé le régulateur 3.6 [V], car le composant qui l’utilisait accepte le 3.3[V] qui est déjà disponible sur la carte.

Nous avons connecté l’ESP32 au microcontrôleur en passant par un des UART (PIO1\_12 TX PIO1\_13 TX).

  
À gauche on peut voir ce qui a été réalisé, mais en testant le module nous avons découvert que l’ESP utilise deux uart (un pour le log et la programmation, le deuxième pour les commandes AT) il faudra refaire la carte en modifiant le schéma comme l’ont fait le groupe de Mathias. Donc pour faire fonctionner la carte déjà réalisée il faut couper les connections des pin 11 et 12 et les re router vers les pins 5 et 6. Il faut aussi connecter la pin 8 au VCC.

Le jumper étant utilisé pour choisir le mode de boot il faut souder un ESP qui est déjà programmé sur notre carte. Vous trouverez la méthodologie de programmation en annexe.

Enfin, pour gagner encore plus de place pour rajouter le module ESP32, nous avons modifié les boutons par d’autres, plus petits.

# Réalisations empreintes

Pour commencer, nous avons réalisé l’empreinte du pont USB-UART. Afin d’éviter le surplus de patte à souder, nous avons rajouté des vias.

Une image contenant texte

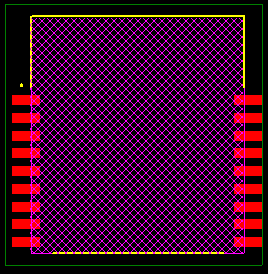
Description générée automatiquement  
*Pont USB-UART*

Même chose pour le régulateur de tension 3.3 [V].

Une image contenant texte, extérieur, trafic, lumière

Description générée automatiquement  
*Régulateur de tension 3.3[V]*

Pour ce qui est du module ESP 32 nous avons augmenté la taille des pads pour faciliter la réparation.

  
*Module ESP32*

Voici les boutons-poussoirs que nous avons utilisés

*Une image contenant texte

Description générée automatiquement  
Bouton-poussoir*

Pour le régulateur de tension, nous n’avons pas mis de via pour avoir suffisamment de soudure pour dissiper la chaleur. Nous avons néanmoins augmenté la taille du pad pour pouvoir y dessouder plus facilement.

Une image contenant texte, moniteur, écran, rouge

Description générée automatiquement  
*Régulateur de tension 5[V]*

# Emplacement et routage PCB

Pour l’emplacement des composants sur la carte, les contraintes suivantes ont dû être respectées :

* La partie RF ne doit pas être modifiée.
* L’écran LCD doit être replacé au centre de la carte.
* Le module ESP32 doit être placé loin de l’alimentation pour éviter des perturbations.

Une image contenant carte

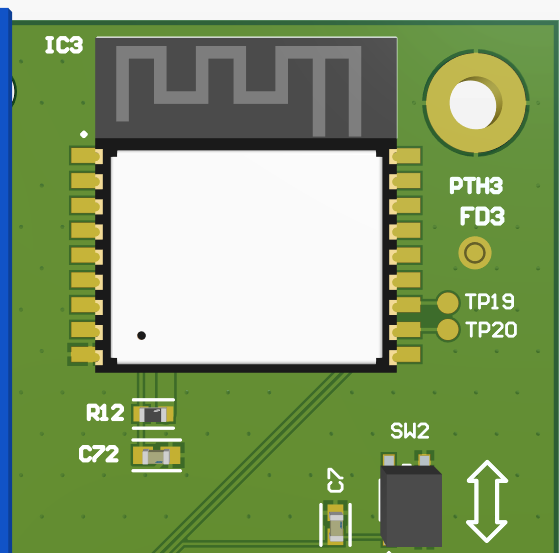
Description générée automatiquement  
*Lecteur RFID*

La partie Alimentation a dû être compressée pour accueillir le nouvel emplacement de l’écran.

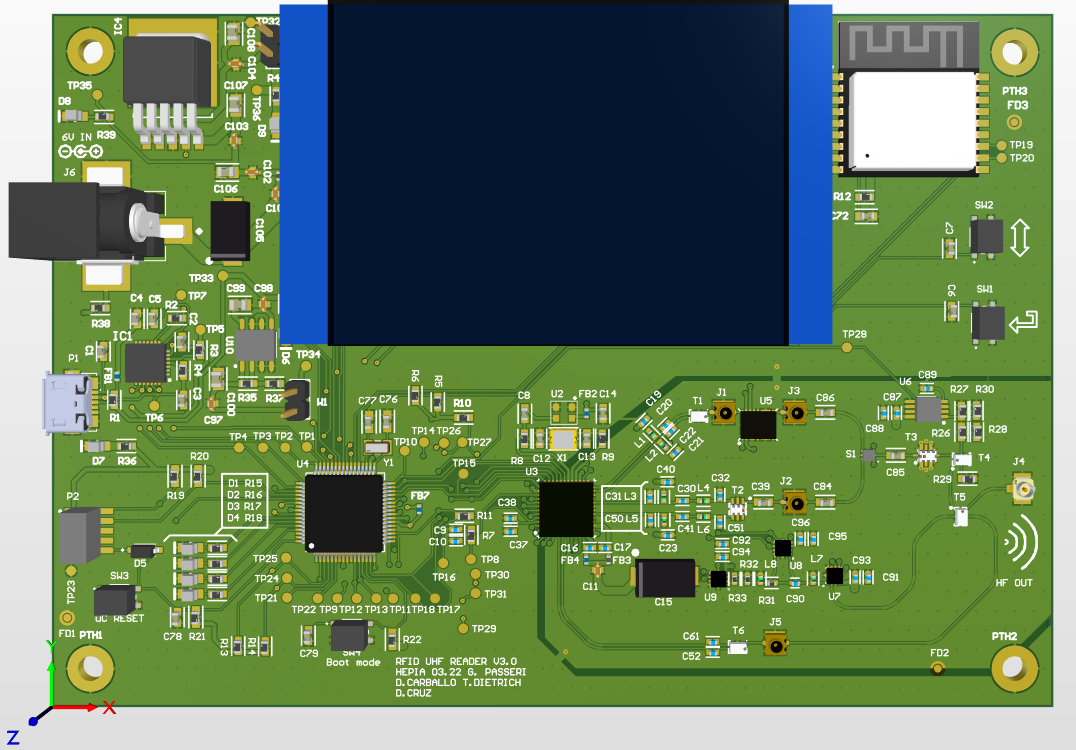
Une image contenant texte, équipement électronique, circuit

Description générée automatiquement  
*Côté gauche de la carte.*

Nous avons placé l’ESP à côté de l’écran, ce qui est le plus éloigné des alimentations.

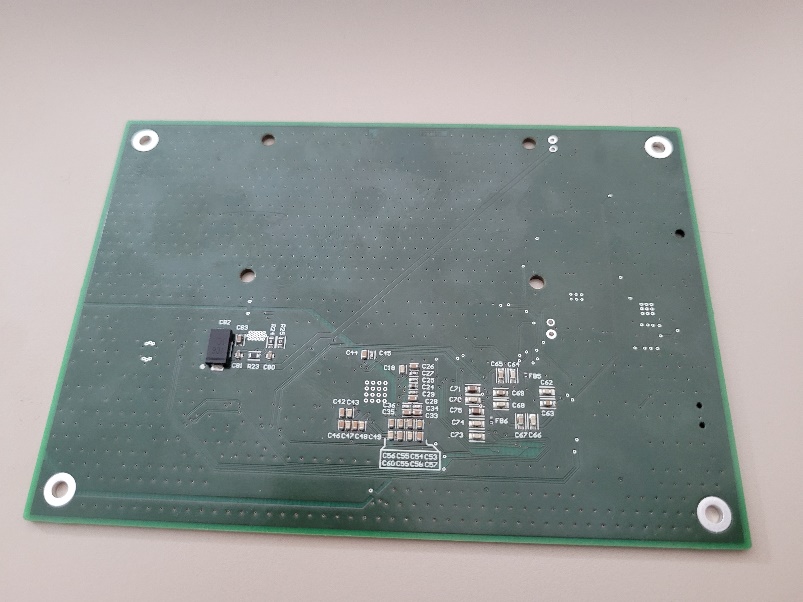
  
*Emplacement ESP32*

Voici donc la carte qui a été envoyée en production.

  
*Vue d’ensemble de la carte*

# Réalisation et test de fonctionnement

Pour ce qui est de la réalisation, le temps étant trop court, nous avons seulement réalisé la face arrière de la carte.



# Conclusion

En conclusion ce qui nous a pris le plus de temps était le routage de la partie alimentation du fait qu’on devait mettre le minimum de composant sur l’autre face. La recherche de composant était aussi difficile, car d’une semaine à l’autre beaucoup de composants n’étaient plus disponibles.

Malheureusement comme écrit auparavant le routage de l’ESP est faux, les erreurs sont réparables, mais il faudrait refaire une version corrigée pour la production finale.

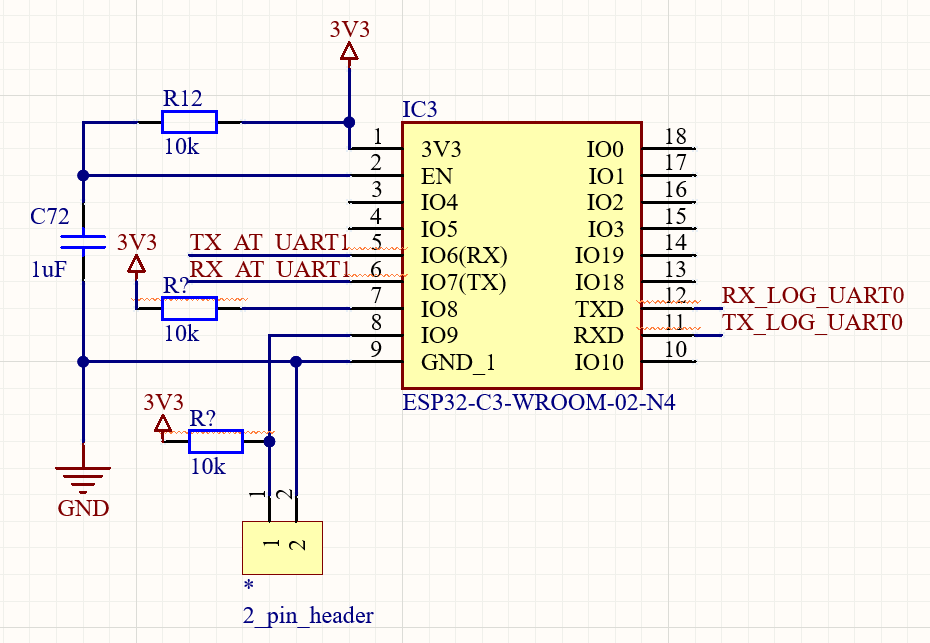
Néanmoins nous avons pu en apprendre plus sur les étapes de conception pour des systèmes embarqués.

# Annexe

## GUIDE ESP32-C3-WROOM-02

Guide réalisé pour cet ESP32 : 356-ESP32C3WROOM02N4

Voici un schéma des connexions à effectuer sur l’ESP32 :



Choix du mode de boot

Jumper = Download

No jumper = SPI

L’UART 0 sert à flasher l’ESP32 et à récupérer les LOG, alors que l’UART 1 permet de communiquer en utilisant les commandes AT une fois que le programme est flashé.

Ces informations viennent de ce tableau disponible dans le guide de l’ESP32.

Une image contenant table

Description générée automatiquement

Une image contenant table

Description générée automatiquementEnsuite l’ESP32 a plusieurs modes de démarrage :

Il y a le mode Download qui permet de flasher l’ESP32 avec son firmware, et le mode SPI boot qui permet de démarrer sur le firmware préalablement flashé.

À noter que les pins IO8 et IO9 ne doivent pas être mise à 0 en même temps.

Pour commencer, coupez l’alimentation de votre ESP32. Connectez la pin IO8 à 1, et IO9 à 0 afin de la mettre en mode download (branchez le Jumper). Connecter votre adaptateur USB-UART sur l’UART0, la communication UART est de 115200 Bauds. Pour finir allumer l’ESP32 en l’alimentant.

Vous devriez obtenir ce message : (Ou n’importe quel message d’erreur, ex invalid\_header)

Une image contenant texte

Description générée automatiquement

Cette étape permet simplement de vérifier que la connexion est fonctionnelle.

Télécharger l’outil esptool à cette adresse :

<https://github.com/espressif/esptool>

Ensuite, ouvrez un terminal dans le dossier, et exécutez la commande suivante :

> esptool.py --port /dev/ttyUSB0 erase\_flash

Voici ce que vous devriez obtenir :

Une image contenant texte

Description générée automatiquementMaintenant on va télécharger le firmware de l’esp32, rendez-vous à cette adresse :

<https://github.com/espressif/esp-at/releases/tag/v2.3.0.0_esp32c3>

Une image contenant texte

Description générée automatiquementCliquez sur ESP32-C3-MINI-1\_AT\_Bin\_V2.3.0.0.zip

Ouvrez le zip, et naviguez dans le dossier ESP32-C3-MINI-1-V2.3.0.0 voici les fichiers qu’il contient :

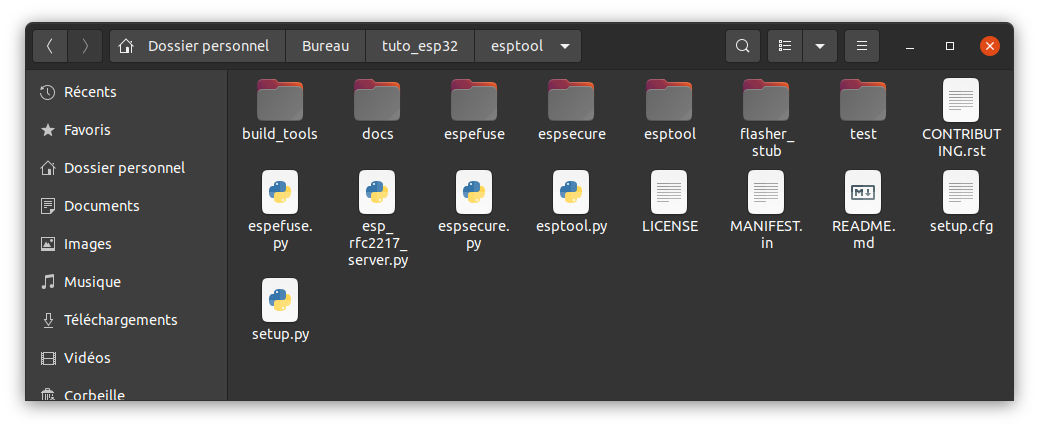
Une image contenant texte, moniteur, noir, capture d’écran

Description générée automatiquement

Décompresser le contenu de ce dossier dans le dossier esptool.

Voici le dossier avant la manipulation :

Une image contenant texte, ordinateur, portable, noir

Description générée automatiquementet après :

Avant de continuer, éteignez et rallumez la carte.

Ensuite, lancez la commande :

> esptool.py --chip auto --port /*dev/*ttyUSB0 --baud 115200 --before default\_reset --after hard\_reset write\_flash -z download.config

**En remplacent download.config avec le contenu du fichier, dans mon cas :**

> esptool.py --chip auto --port /*dev/*ttyUSB0 --baud 115200 --before default\_reset --after hard\_reset write\_flash -z --flash\_mode dio --flash\_freq 40m --flash\_size 4MB 0x8000 partition\_table/partition-table.bin 0xd000 ota\_data\_initial.bin 0xf000 phy\_init\_data.bin 0x0 bootloader/bootloader.bin 0x60000 esp-at.bin 0x1e000 at\_customize.bin 0x1F000 customized\_partitions/ble\_data.bin 0x3a000 customized\_partitions/mqtt\_key.bin 0x27000 customized\_partitions/server\_key.bin 0x3c000 customized\_partitions/mqtt\_ca.bin 0x2d000 customized\_partitions/client\_key.bin 0x2b000 customized\_partitions/client\_cert.bin 0x31000 customized\_partitions/factory\_param.bin 0x2f000 customized\_partitions/client\_ca.bin 0x38000 customized\_partitions/mqtt\_cert.bin 0x29000 customized\_partitions/server\_ca.bin 0x25000 customized\_partitions/server\_cert.bin

Normalement le programme devrait s’écrire dans la flash de l’ESP32.

Éteignez votre ESP32, puis connecter la pin IO8 et IO9 à 1 afin qu’il démarre en mode SPI (enlever le jumper).

Ensuite pour vérifier que tout s’est bien passé, ouvrez un picocom sur votre ESP32,puis rallumez la carte, voici ce que vous devriez obtenir :

Une image contenant texte

Description générée automatiquement

Si tout c’est bien passé, l’ESP32 démarre bien sur le firmware flashé, il ne reste plus qu’as changer d’UART pour communiquer avec les commandes AT.

Connecter votre convertisseur USB-UART au pin IO6 et IO7

Ouvrez un picocom avec cette commande :

> picocom /dev/ttyUSB0 --baud 115200 --omap crcrlf

Éteignez et rallumer la carte, un ready devrait apparaitre, entrez AT, puis faite ENTER, la carte devrait répondre OK

Une image contenant texte

Description générée automatiquement

La carte est prête à être programmée avec les commandes AT, bonne chance.