UNIVERSITÉ DE SHERBROOKE

Faculté de génie

Département de génie électrique, génie informatique et génie robotique

Documentation   
ESP32 CHUNGUS RAPIDUS REV. 1.1

Groupe technique Robotique UdeS

Biogénius

Présenté aux

Membres de l’équipe Biogénius, Contrôle

Présenté par

Jacob Turcotte – turj3113

Sherbrooke – 24 Mai 2024

Table des matières

[1. Introduction 5](#_Toc492847343)

[2. Développement 5](#_Toc492847344)

[2.1 Exemple d’équation 5](#_Toc492847345)

[2.2 Exemple de tableau 5](#_Toc492847346)

[2.3 Exemple de graphique 6](#_Toc492847347)

[2.4 Exemple d’histogramme 6](#_Toc492847348)

[3. Conclusion 6](#_Toc492847349)

[4. Références 8](#_Toc492847350)

[Annexe A Exemple d’annexe 9](#_Toc492847351)

[Annexe B Exemple d’annexe 10](#_Toc492847352)

Liste des figures

[Figure 1: Exemple de graphique 6](#_Toc492847353)

[Figure 2 : Exemple d'histogramme 6](#_Toc492847354)

Liste des tableaux

[Tableau 1: Exemple de tableau 5](#_Toc492847355)

Lexique

|  |  |
| --- | --- |
| Terme technique | Définition |
|  |  |
|  |  |

Liste des symboles

|  |  |
| --- | --- |
| Symbole | Définition |
|  |  |
|  |  |
|  |  |

Liste des acronymes

|  |  |
| --- | --- |
| Acronyme | Définition |
|  |  |
|  |  |
|  |  |

# Introduction

Ce document a été conçu pour vous guider à travers les différentes étapes et composants de notre système, depuis l'identification des problèmes courants jusqu'aux pistes d'amélioration futures. Vous trouverez des sections détaillées sur la connexion des différents modules, l'alimentation, et l'utilisation de périphériques I2C tels que le multiplexeur, les extensions GPIO et PWM. Nous couvrirons également l'intégration du microcontrôleur ESP32, ainsi que les solutions d'affichage à LED. Chaque partie est structurée pour offrir une compréhension claire et pratique, facilitant ainsi la mise en œuvre et le dépannage.

# Problèmes connus et solutions

## Liste de problèmes

* Alimentation 3v3 qui n’allume pas
  + Si nous allons voir la documentation des MIC5504-3.3YM5 (REF), à la page 9 (REF image). Il est expliqué que la pin ENABLE doit être à HIGH (tension plus haute que 1.2V). Bien qu’il ait une résistance pull-down, il est préférable de mettre ce pin à HIGH pour assurer le bon fonctionnement de l’alimentation.

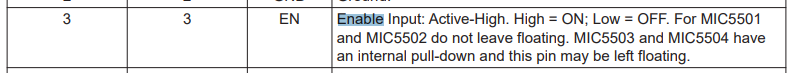


Figure 1: Page 9 documentation MIC5504

* + La REV1.1 a un saut de soudure disponible sur chacun des MIC5504 mais celui -ci va à la terre (REF image). Ainsi, l’utilisation du saut de soudure ne règle en aucun cas le problème.

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Figure 2: Démonstration du problème d'alimentation

* Sérigraphie à l’envers
  + La sérigraphie des connecteurs du côté gauche (connecteur pour le BNO de la hanche gauche et connecteur du MD gauche), sont tous les deux inversé de 180 degrés (REF image).

Une image contenant texte, capture d’écran, cercle, Police

Description générée automatiquement

Figure 3: Démonstration de la mauvaise sérigraphie du connecteur pour MD gauche

* Manque le connecteur TTL pour le HMI
  + Le connecteur pour le HMI est manquant sur le PCB. Les pins à utiliser sont, GPIO17 et GPIO16. Il manque aussi les connections nécessaires pour alimenter le HMI.
* Reset des capteurs sur pin GPIO TX
  + La pin GPIO17 utilisée comme transmetteur (TX) afin de communiquer avec le HMI est déjà utilisé (REF image). Elle est connectée au signal Reset envoyé aux capteurs. Cela a été fait afin de pouvoir mettre le signal à LOW dans le code si jamais le MCU détecte une erreur venant des capteurs.

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Figure 4: Démonstration de l'erreur RESET dans le schéma électrique

* Connection I2C sur le module PWM inversé
  + Les pins de la connexion I2C allant du MCU au PCA9685PW sont inversé. Le SDA du Esp32 est dans le SCL du module et le SCL du MCU est dans le SDA du PCA9685PW.
* PULLUP Nécessaire sur la moustache
  + Les proxims sont trop éloignés et le SCL n’a pas le temps de revenir à 3v3. Le fix est d’ajouter une PULLUP 2k sur la moustache car en ce moment c’est la pullup du BNO qui s’en occupe qui est à genre 3m.

Une image contenant texte, diagramme, Plan, carte

Description générée automatiquement

Figure 5: Démonstration du schéma électrique avec erreur I2C

* Connecteur du relais et du BNO dos sont trop proche
  + Les deux connecteurs sont beaucoup trop proches de l’un et l’autre. Dès que l’on branche les fils dessus, il n’est pas possible de brancher l’autre sans que celui-ci soit incliné (REF image).

Une image contenant capture d’écran, Police, Appareils électroniques, texte

Description générée automatiquement

Figure 6: Démonstration dans le PCB editor des connecteurs trop proches

* BNO dos fait tout planter
  + Lorsque branché au PCB principale, le BNO dos fait tout planter. La communication I2C ne marche plus et aucun autre BNO est détecté.
* Pas capable de télécharger du code dans le ESP32
  + La pin GPIO12 doit être à LOW lorsque nous téléchargeons du code dans le ESP32. La pin en question est branché à l’encodeur du moteur genou droit, il se peut que la pin soit à HIGH à certains moments.

## Liste de solutions temporaires faits

* Alimentation 3v3
  + Nous avons utilisé un fil de cuivre émaillé de 4oz afin de souder les pins ENABLE au 5V entrant (REF image).

Une image contenant Appareils électroniques, circuit, Ingénierie électronique, Composant électronique

Description générée automatiquement

Figure 7: Démonstration de la solution temporaire de l'alimentation

* Sérigraphie
  + Nous avons utilisé un marqueur noir afin d’effacer la mauvaise sérigraphie et de réécrire le bon brochage (REF image).

Une image contenant circuit, Ingénierie électronique, texte, capture d’écran

Description générée automatiquement

Figure 8: Démonstration de la sérigraphie temporaire

* Connecteur HMI
  + Nous avons soudé un connecteur Molex nano fit 2 par 2 sur les pins GPIO17 (TX), GPIO16 (RX) et nous avons utiliser les points de tests 5V et GND afin d’alimenter le HMI (REF image).

Une image contenant Appareils électroniques, circuit, Ingénierie électronique, Composant électronique

Description générée automatiquement

Figure 9: Démonstration du connecteur 2 par 2 HMI

* GPIO17
  + Nous avons gratté chaque trace RESET allant au GPIO17. Il a ensuite été nécessaire de reconnecter toutes ces traces ensemble afin de s’assurer du bon fonctionnement du bouton RESET. Nous avons utilisé un fil de cuivre émaillé de 4oz afin d’y arriver (REF image).

Une image contenant Appareils électroniques, circuit, Ingénierie électronique, Composant électronique

Description générée automatiquement

Figure 10: Démonstration du fixe du GPIO17

* I2C Module PWM
  + Nous avons dessoudé les pins I2C du module et nous les avons resouder correctement avec l’aide de fils de cuivre émaillé 4oz (REF image).

Une image contenant Appareils électroniques, circuit, Ingénierie électronique, Composant électronique

Description générée automatiquement

Figure 11: Démonstration de la reconnexion du port I2C au module PWM

* Connecteur relais et connecteur BNO dos
  + Nous avons sablé l’un des 2 connecteurs afin de laisser assez de place pour l’autre (REF image).

Une image contenant câble, fils électriques, machine, Tubes thermorétrécissables

Description générée automatiquement

Figure 12: Démonstration des connecteur BNO dos et relais

* BNO dos
  + Le problème ne vient pas du PCB CHUNGUS RAPIDUS, il vient du boitier créer afin de protéger le BNO. L’une des 2 vis rattachant le boitier au frame était trop haute et causait un court-circuit entre le GND, RESET, SDA et SCL. Le fix a été de rajouter des rondelles afin d’augmenter la distance du PCB BNO et l’enveloppe. Nous avons aussi changé la vis M6 pour une vis M3.
* Télécharger du code dans le EPS32
  + La solution temporaire est simplement de faire tourner le moteur genou droit jusqu’à temps que celui-ci soit un niveau LOW.
* PULLUP nécessaire pour les moustaches
  + Nous avons soudé une 2k entre SDA et 3v3 puis SCL et 3v3. Nos signaux sont beaux à nouveau

# Liste des connections

## Connections vers capteurs ou actuateurs (REF image)

* Une connexion BNO dos 2 par 4 (rouge)
* Une connexion relais 2 par 4 (vert)
* Deux connexions « Motor Drive » 2 par 5 (jaune)
* Deux connexions pour les encodeurs des hanches (bleu)
* Deux connexions aux moustaches 2 par 8 (mauve)

Une image contenant texte, circuit, capture d’écran, Ingénierie électronique

Description générée automatiquement

Figure 13: Connections capteurs/actuateurs

## Connections pour ESP32

* Se référer à la section (REF section ESP32) pour plus de détails sur le pin out. (Jaune)

Une image contenant texte, circuit, capture d’écran, Ingénierie électronique

Description générée automatiquement

Figure 14: Connexion ESP32

## Connections pour alimentation

* Il a deux façons d’alimenter le PCB
  + La première en rouge, est d’utiliser le connecteur « Barrel jack »
  + La deuxième en jaune, est de souder le GND au trou carré et le 5V au trou rond.

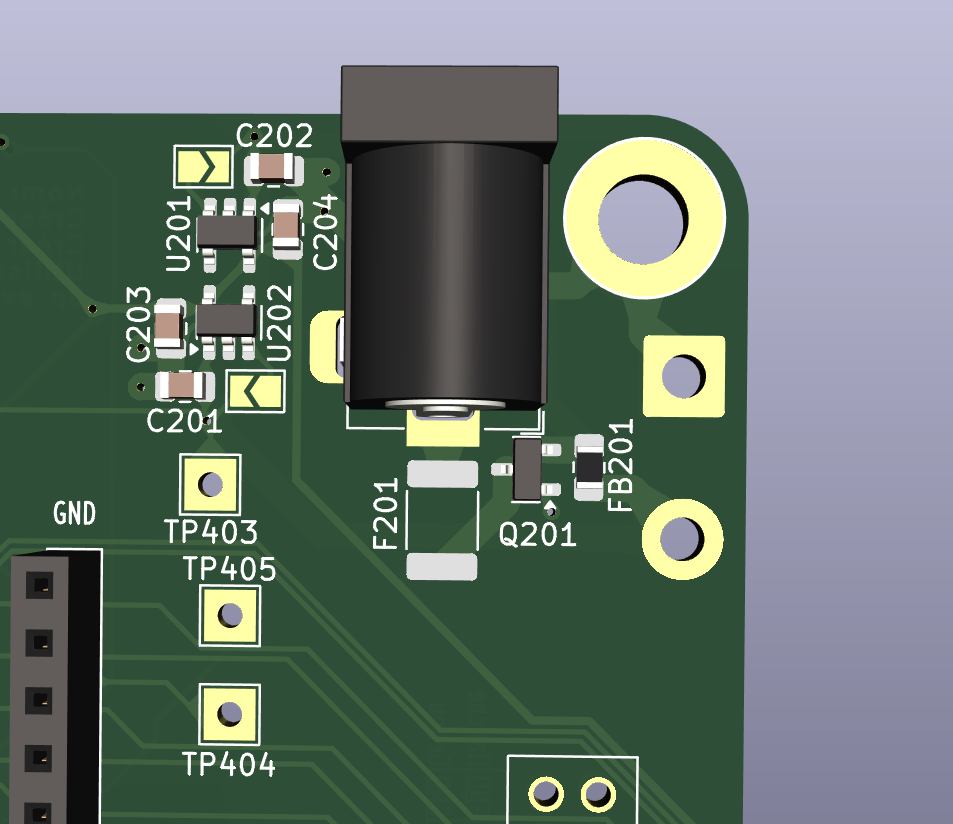


Figure 15: Connections alimentation

# Alimentation

Il a 3 alimentation externes possible au circuit « Chungus rapidus ». Il a celle USB venant du ESP32, le « Barrel jack » et les fils que nous pouvons souder. Chacune de ces alimentations sont 5V.

Il a au total, 3 régulateurs de tension sur le circuit. Deux MIC5504-3.3YM5-TR (régulateur 3v3) et un régulateur 3v3 sur le ESP32 (AMS1117-3.3). Au total, le circuit peut alimenter 1,6A à 3v3 ce qui est plus que suffisant pour alimenter tout les circuits et capteurs (environs 750mA total).

# TCA9548 (Multiplexeur I2C)

L’adresse I2C du multiplexeur est le 0x70. Le multiplexeur permet de rediriger la communication I2C d’un port vers 8 autres ports I2C. Nous avions un problème d’adresse où les BNO ont juste 2 adresses possibles. Avec 5 BNO sur l’exosquelette, on avait des erreurs de communication tout le temps. Le TCA8548 vient régler se problème. Nous utilisons présentement seulement 3 des 8 ports I2C disponibles.

Voir I 2C-scanner dans le GitHub pour plus d’information.

# PCA9685 (PWM)

L’adresse I2C du PCA9685 est le 0x7C. Il permet de diminuer les pins utilisés du ESP32 et d’alléger le code. Les PWM sont utilisés afin de contrôler la vitesse et la puissance des moteurs. Au total, nous utilisons 8 des 16 pins PWM disponible sur le PCA9685.   
L’adresse de celui-ci peut être changé au besoin en soudant les ponts d’étain.

Voir ESP-Motor dans le GitHub pour plus d’information.

# PCA9505 (GPIO)

L’adresse I2C du PCA9505 est le 0x20. Il permet aussi de diminuer les pins utilisés du ESP32. Cependant celui-ci n’allège pas le code. Le PCA9505 est un « GPIO extender », il rajoute 40 pins GPIO à un circuit. Dans notre cas nous utilisons seulement les pins comme sortie car le protocole I2C n’est pas assez rapide pour gérer les encodeurs.   
Sur les 40 pins, 17 sont utilisées. L’adresse de celui-ci peut aussi être changé au besoin.

Voir Pin\_extender dans le GitHub pour plus d’information.

# ESP32

Une image contenant texte, capture d’écran, nombre, Parallèle

Description générée automatiquement

Figure 16: Tableau du pin out du DevBoard WROOM ESP32

Nous utilisons le ESP32 comme microcontrôleur pour sa versatilité et sa vitesse d’exécution. Son coût abordable est un gros plus aussi. Nous avons décidé d’utiliser un DevBoard afin de nous faciliter la tâche et sauver du temps pendant la conception du circuit. S’il vous plaît vous référer à la figure (REF) pour le pin out.

# DEL

# Points de tests

# Références

**There are no sources in the current document.**

1. MIC5501



1. MIC5501



1. Exemple d’annexe