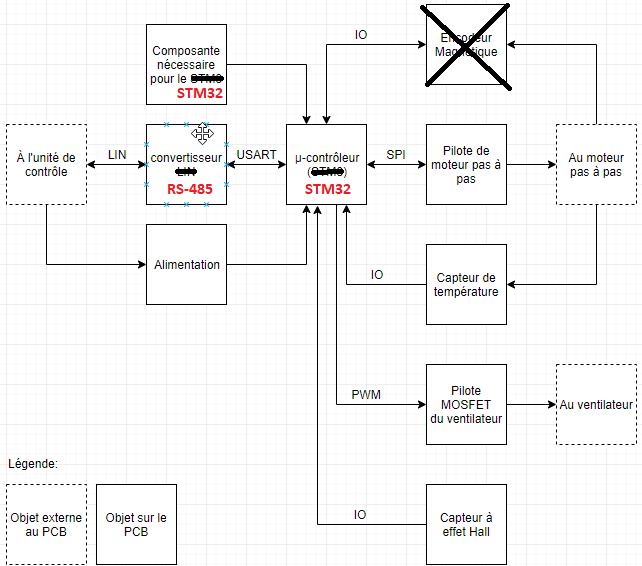
# Démarrage

## Architecture Matérielle :



## Planification :

### Semaine 1 et 2 :

* Révision des objectifs
* Planification
* Révision des PCB
* Setup IDE
* Setup gestion fichiers
* Code des pilotes

### Semaine 3 :

* Dessouder les Blue Pills
* Assemblage 2 PCB
* Debug des PCB
* Test des pilotes

### Semaine 4 :

* Debug des PCB
* Test des pilotes

### Semaine 5 :

* Assemblage de tous les PCB (Semaine 3 & 4)
* Montage des PCB sur le bras
* Mécanique du bras final

### Semaine 6, 7 et 8 :

* Programmation objets

### Semaine 9 :

* Interface GUI python
* Intelligence du bras? (Python / C++)
* Intégration

### Semaine 10 :

* Intégration
* Rapport

## Livrables :

* 6 PCB de pilotes des moteurs.
* 1 PCB d’interfaçage des périphériques du BeagleBone Black.
* 6 axes du bras robotisé.
* 1 interface graphique pour l’écran LCD.
* 1 programme de contrôle des moteurs.
* 1 programme de gestion des capteurs.
* 1 programme de gestions des erreurs.
* 1 programme de communication ~~LIN~~ RS485.
* 1 programme de contrôle intégral du bras robotisé.
* 1 schéma des pilotes des moteurs
* 1 schéma de l’interfaçage des périphériques du BeagleBone Black.
* 1 circuit d’alimentation fonctionnel dans le système.
* ~~1 interface graphique pour le PC (bonus)~~
* 1 rapport complet
* Acheter le matériel nécessaire à la mise en marche du bras.

## Microcontrôleur :

Le choix de notre microcontrôleur s’est arrêté sur le STM32F103C8T6. Il s’agit d’un microcontrôleur de le famille STMicroelectronics. Il nous permet d’interfacer tous nos périphériques sans problème. Tant bien, qu’il y a des GPIOs de libre. Le package est standard (Quad Flat Package), ce qui nous permettrait de changer de microcontrôleur sans trop redesigner notre PCB.

# Environnement de travail

## Organisation :

Notre structure de projet était d’abord située dans un folder sur Google Drive. L’idée n’était pas mauvaise au départ, puisqu’il n’y avait que de la paperasse à faire. Ainsi, il était plus simple de simplement créer des fichiers .gdoc que de rendre les choses plus complexes dans un projet git. Une fois le projet entamé on a vite réaliser qu’un projet sur google drive peut vite devenir chaotique. C’est pourquoi on a migré vers git. La gestion de sources s’est vue beaucoup simplifié sur le long terme.

## IDE :

Notre choix d’IDE s’est arrêté sur VSCode. Nous voulions utiliser un environnement «user friendly» et ne pas perdre trop de temps à paramétrer notre environnement de travail. Ainsi, VSCode avec l’ajout de l’extension platform.io permettent de coder rapidement évitant de cette manière de passer trop de temps à faire fonctionner l’environnement et non le code.

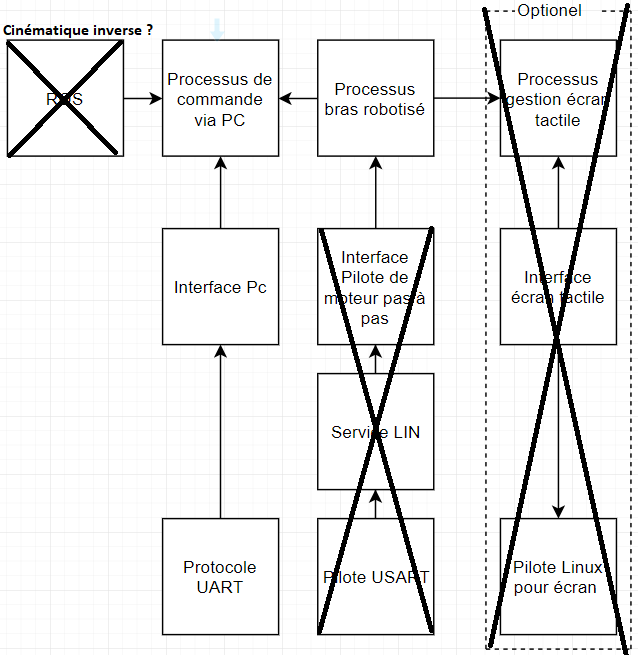
# Conception électronique

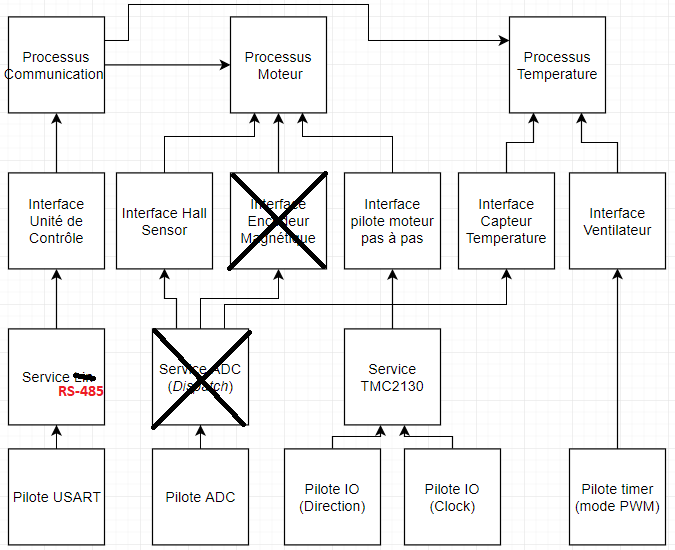
## Choix des pièces :

Il y avait beaucoup de pièces à trouver. Pour les grosses composantes, il s’agissait de trouver ce qui coûtait le moins cher, mais qui allait être de qualité. Donc, pour le TMC2130 et le STM32, il n’y avait pas vraiment de compromis à faire. Les composantes passives ont été choisies en fonction des datasheets des composantes principales (ce qui était recommandé). Il y a plusieurs composantes qu’on a simplement pu récupérer de ce qu’on avait déjà. Nous avions à notre disposition des MAX485 et le «power supply».



## Architecture logicielle :





# Préparation

La librairie de pièces comprend toutes les pièces du projet. Elle est liée à un document Excel, permettant ainsi d’identifier chaque composante à un identifiant unique. Voici un aperçu :



# Dessin et programmation

## Schémas :

Les schémas du pilote de moteur sont sur 2 pages et il y a une page de blocs hiérarchique. Celui du BeagleBone Black est sur 1 page.

\*\*IMAGE SCHÉMAS\*\*

## Fonctionnement des pilotes :

\*\*CODE?\*\*

# Conception mécanique :

Le PCB du pilote de moteur représentait un défi en soi. Il y avait une contrainte d’espace sur les axes du bras assez limitante. Notre PCB devait faire moins de 4.2cm de diamètre et en plus devait être circulaire. Le «routing» s’est avéré plutôt difficile, mais nous avons réussi à faire un placement des composantes logique et fonctionnel.

\*\*IMAGE PCB 3D et NORMAL\*\*

# Correction et intégration

## Assemblage :

# Vérification et validation

## Test des logiciels :