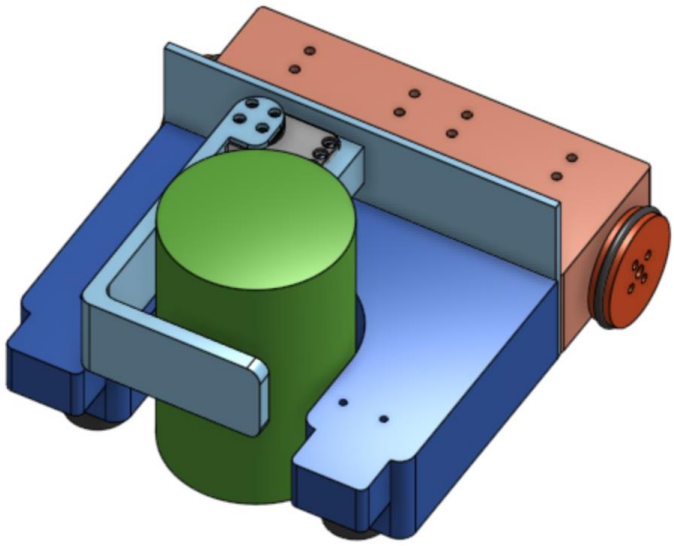
Rapport de projet de 3ème année option ESE

Robot bière

Encadrants

- Mr FIACK Laurent
- Mr TAUVEL Antoine



COMLAN Chelsea - KOEHL Loïcia - HAVRET Alix - SIMON Quentin

Novembre 2022

ENSEA

Beyond Engineering

1 Introduction

1.1 Présentation et objectif du défi “Robot Bière"

Élèves en option ESE (Electronique et Systèmes Embarqués) au sein de l'ENSEA (Ecole Nationale Supérieure de l'Electronique et de ses Applications), réalisant un robot comme projet central de leur option.

Création d'un robot permettant d'attraper des canettes et les déplacer dans des zones, délimitées aux préalables, selon la couleur de la canette.

Le robot devra être capable de :

- Se déplacer
- Saisir une canette
- Se déplacer avec la canette
- Détecter la couleur de la canette
- Ne pas tomber
- Déplacer la canette dans la zone correspondante

1.2 Analyse fonctionnelle /CDC

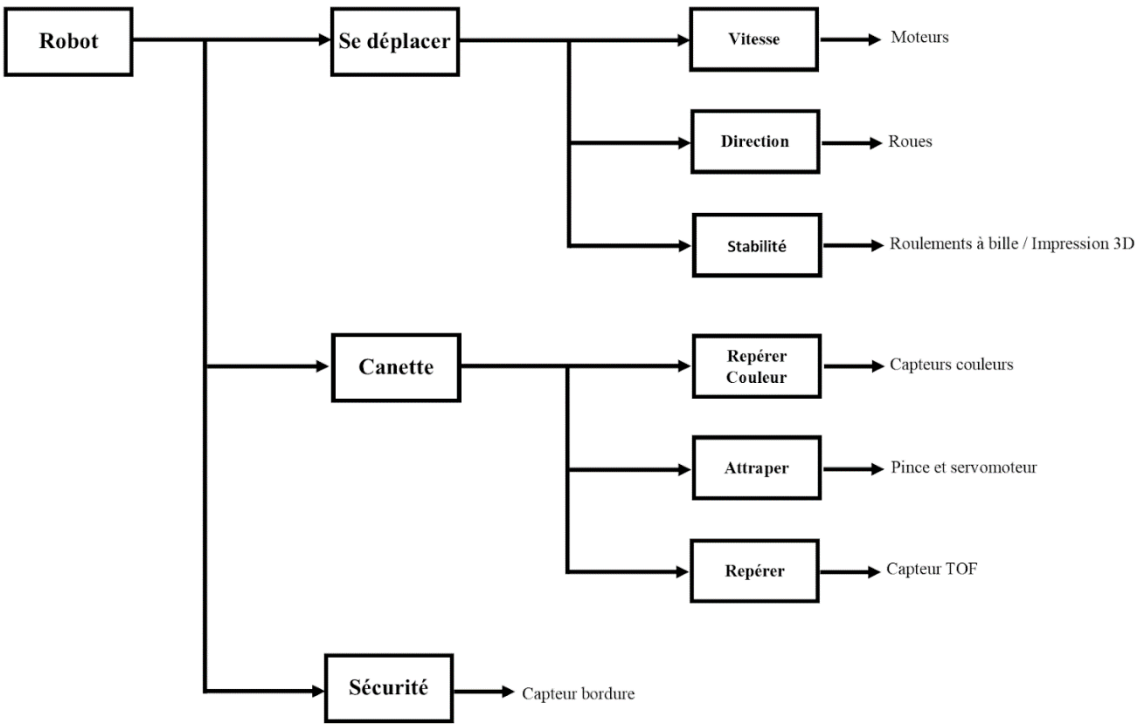


Figure 1 Analyse fonctionnelle

1.3 Organisation du projet

1.3.1 GitHub

Github projet

1.3.2 Equipe

- Chelsea COMLAN : responsable hardware
- Loïcia KOEHL : responsable intégration
- Quentin SIMON : responsable software
- Alix HAVRET : responsable livrable

1.3.3 Planning

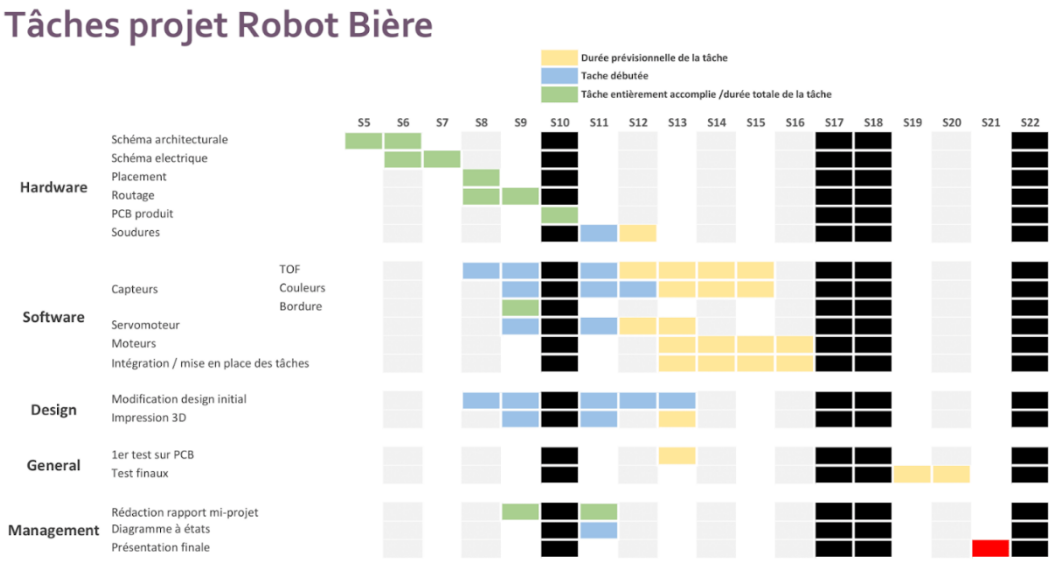


Figure 2 Gantt projet robot bière

2 Architecture du projet

2.1 Ressources

2.1.1 Matérielles

Nom du composant	Référence
STM32G070RBT6	Farnell : 3365377
Quartz 16 MHz	Farnell : 2853935
Connecteur SWD/ST Link	Farnell : 3226055
Driver moteur : ZXBM5210-S	Farnell : 3935372
Moteurs FIT 0520	DfRobot FIT0520
Régulateur 5V AZ1117CH2-5.0TRG1	Farnell : 3942512
Régulateur 3.3V MIC5317-3.3YM5	Farnell : 2920683
Batterie NIMH 7.2V 1.3Ah	RS : 777-0377
Capteur bordure	Sharp GP2Y0D805Z0F
Capteur TOF	VL53L0X
Capteur de couleur 5V	SEN0101
Capteur de couleur 3.3V	VEML3328
Servomoteur : XL320	Generationrobots : A-000000-01282
Connecteurs JST 2.54mm	
LED + R/C	

Pour notre projet, nous aurons également besoin du matériel de laboratoire : oscilloscope, générateur, multimètre, station de soudures

2.1.2 Logicielles

Partie Software
- Cube IDE

PCB
- Kicad pour la réalisation
- JCLPCB pour la production

Design du robot
- On Shape pour la réalisation
- ENSEA pour la production

3 Étapes du développement

3.1 Hardware

Pour structurer notre carte électronique nous avons dans un premier temps réalisé un schéma architectural (cf Annexe 1)

Ce schéma recense l'ensemble des composants nécessaires au bon fonctionnement de notre robot, ainsi que l'ensemble des ports sollicités.
La visualisation de l'ensemble des ports nécessaires a été faite sur CubeIDE avec le microcontrôleur correspondant (cf Annexe 2).

Sur Kicad nous avons traduit le schéma architectural, avec l'ensemble des composants, et broches nécessaires. Par la suite, le placement et le routage a été réalisé pour être contenu sur un PCB 4 couches de dimensions 11 par 4.

Une fois les PCB reçus, la soudure a débuté.

3.2 CAD

En parallèle de la conception du PCB, intéressons-nous à la forme que prendra notre robot. Nous partons du design proposé sur OnShape auquel nous ajoutons ou modifions certaines parties.

Pour assurer la stabilité du robot nous voulons qu'il compte deux roues folles à l'avant, de part et d'autre du creux pouvant accueillir la canette. Pour ce faire, nous devons déplacer le servomoteur qui commande le clapet de la pince afin de placer de manière symétrique les deux roues folles. La pince prend une forme différente, fixée au fond du creux qui accueille la canette.

Afin de placer et fixer les différents capteurs sur le robot, nous créons des ouvertures ainsi que des parois de fixation liées aux dimensions des différents composants comme pour les capteurs TOF qui seront placés à l'avant du robot, le capteur couleur au fond du creux pour la canette et le servomoteur juste au-dessus.

Une fois toutes les pièces modifiées, nous envoyons le design pour impression 3D et attendons que les pièces de notre robot soient fabriquées pour pouvoir l'assembler puis le tester voire l'améliorer.

3.3 Software

3.3.1 Capteur de bordure

Pour détecter les bordures de la table, on utilise un capteur de bordure Sharp. Ce dernier sera positionné en amont du robot et des roulements à bille, en pointant vers la table.

Figure 3 Capteur de bordure

Il envoie un rayon laser pour détecter si un objet se trouve à une distance inférieure à 5 cm. Si c'est le cas, la sortie est à l'état bas. Sinon, la sortie est à l'état haut.

Le capteur est alimenté avec le régulateur 5V.
Son fonctionnement a été testé avec la carte STM32F746NG.

Ainsi, pour empêcher une chute, on branche la sortie du capteur sur un GPIO en mode EXTI. Puis on active une interruption à chaque front montant.

Putty nous permet d'observer le bon fonctionnement du code.

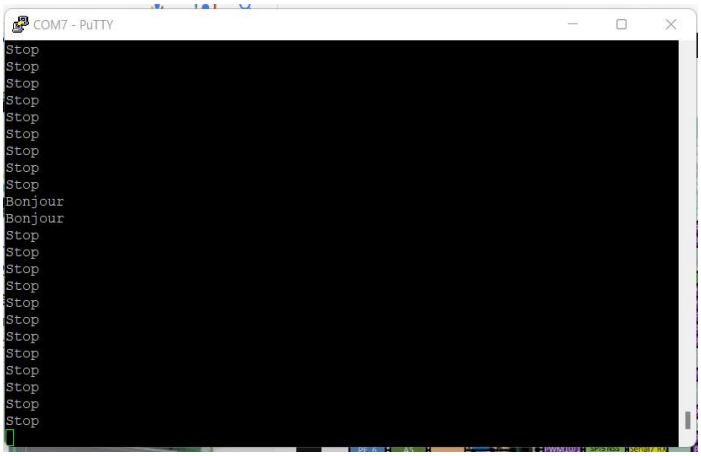


Figure 4 Sortie de la console TeraTerm pour l'utilisation du capteur de bordure

A terme, le capteur sera placé en amont du robot. Si le vide est détecté, le message d'erreur sera remplacé par une commande d'arrêt moteur, suivi d'une commande de rotation.

On remarque qu'on obtient bien le changement d'état en parallèle à l'oscilloscope.

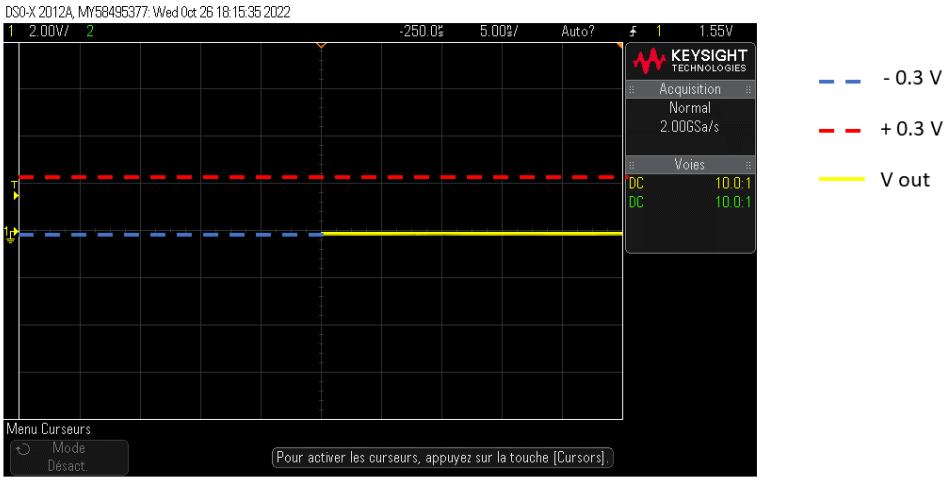


Figure 5 Chronogramme de la sortie du capteur avec un obstacle à moins de 5 cm

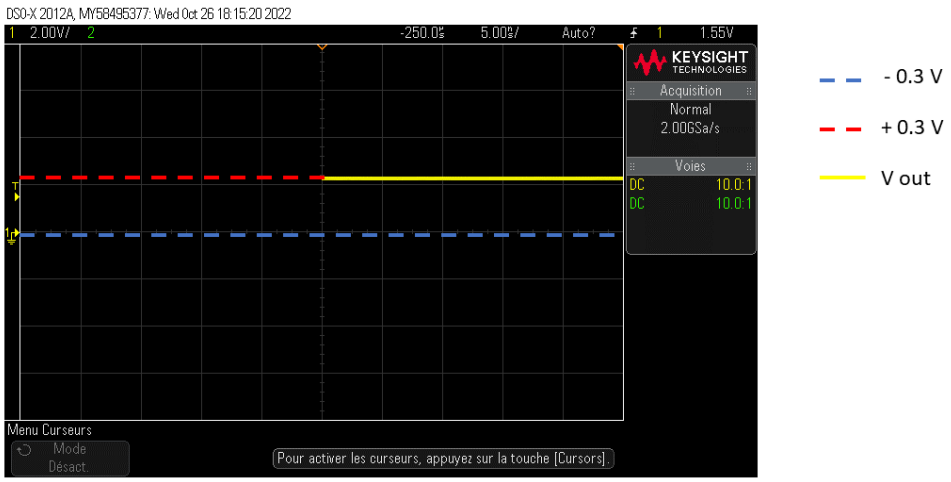


Figure 6 Chronogramme de la sortie du capteur avec un obstacle à plus de 5cm

3.3.2 Capteur couleur VEML3328

Pour détecter la couleur des canettes nous allons utiliser le capteur de couleur VEML3328.

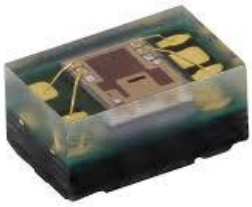


Figure 7 Capteur couleur VEML3328

Ce **capteur mesure la valeur de bleu, rouge, vert et IR d'un objet**. Il stocke les valeurs dans des registres sur 16 bits. Le microcontrôleur peut accéder à ces valeurs par liaison I2C, grâce à l'adresse du capteur 0x10 et les codes de commande, 0x05 pour le rouge et 0x06 pour le vert.

Avant toute chose, il faut configurer le capteur grâce au code de commande 0x00.

Cependant, il fonctionne avec une tension d'entrée de 3.3V contrairement à l'autre composant de la carte. C'est pourquoi nous avons réalisé un autre PCB pour le capteur.

<u>Compositions du PCB :</u>

- 4 diodes et 4 résistances pour éclairer la canette
- Un régulateur 3.3V pour alimenter le capteur
- 2 résistances de pull-up pour la communication I2C
- 3 condensateurs de découplage pour le régulateur et le capteur

Cf Annexe 3.

3.3.3 Capteur couleur SEN0101

Nous utiliserons un second capteur de couleur. Le SEN0101 de chez DFRobot comprend une puce capteur couleur et 4 LEDs blanches.

Figure 8 Capteur couleur SEN0101

Il peut être alimenté en 3.3V ou 5V, on l'alimente par le régulateur 5V. L'utilisation du capteur est testée avec une carte STM32F746NG.

Dans un premier temps, les 4 LEDs blanches sont allumées en mettant à 1 le signal envoyé sur la pin LED du capteur. Le capteur est bien alimenté.

Pour faire fonctionner le capteur, les pins S0, S1, S2, S3 ainsi que le port LED sont branchés sur des ports GPIO_Output. La liaison UART qui passe par le port USB de la carte est elle aussi configurée pour avoir des retours pendant que le code tourne sur la carte afin de débugger. Le but est, à terme, **d'afficher la couleur détectée.**

Les pins S2 et S3 permettent de choisir le filtre pour détecter une couleur en particulier entre rouge, vert, bleu. Il est également possible de configurer le capteur sans filtre. Les pins 20 et S1 permettent de choisir l'échelle d'intensité lumineuse qui est reliée à la fréquence en sortie du capteur. Dans un premier temps, l'échelle de sortie en fréquence est à 100%.

S0	S1	OUTPUT FREQUENCY SCALING (fo)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

S2	S3	PHOTODIODE TYPE
L	L	RED
L	H	BLUE
H	L	Clear (no filter)
H	H	GREEN

Figure 9 Table de fonctionnement du capteur couleur et des LEDS

La sortie Out récupère une fréquence qui permet de déterminer la couleur de la canette. La pin Out du capteur est reliée à **un timer en mode Input Capture** afin de mesurer la **fréquence du signal** qui sera capté par les photodiodes. Le capteur envoie un signal carré au STM et pour déterminer la fréquence du signal il faut faire la différence entre les temps des fronts montants sur une ou plusieurs périodes.

Mode de fonctionnement des mesures :

Pour lancer la première mesure avec le mode input capture, on utilise HAL_TIM_IC_Start_IT. Selon la configuration choisie, les différentes mesures sont effectuées sur les fronts montants ou descendants ou les deux et les valeurs sont stockées dans le registre IRQ_CCR. Plusieurs mesures permettent de faire une moyenne qui donnera un résultat plus fin de la fréquence et donc de la couleur de la canette. Après avoir effectué plusieurs mesures, il faut stopper les interruptions IC avec la fonction HAL_TIM_IC_Stop_IT.

A chaque interruption, le code appelé est dans la fonction HAL_TIM_IC_CaptureCallback. Le capteur est configuré pour recueillir deux temps de montée consécutifs et en faire la différence pour obtenir la période puis en déduire la fréquence grâce à l'horloge du timer.

A chaque interruption, un compteur est incrémenté pour faire un certain nombre de mesures qui seront moyennées et aussi pour changer le type de filtre. Pour debugger, des printf ont été utilisés dans le code d'interruption ce qui n'est pas recommandé et peut être amélioré. Pour l'heure, des fréquences sont affichées en sortie mais ne semblent pas cohérentes et il reste à moyenner les fréquences et à les faire correspondre aux couleurs pour vérifier la capacité du capteur à différencier et classer les canettes qui seront vertes et rouges.

3.3.4 Capteur TOF VL53L0X

Afin de pouvoir **repérer les canettes**, et **la distance du robot vis à vis de ces dernières**, nous utilisons un capteur TOF (Time Of Flight). Ce dernier relève la distance en cm (il nous offre une plage de mesure de distance jusqu'à 2 mètres) du capteur par rapport à un objet, quelle que soit sa couleur.

Figure 10 Capteur TOF VL53L0X

C'est la mesure du temps que met la lumière à revenir au capteur qui permet de déterminer sa distance aux objets

Le VL53L0X de chez STMicroelectronics est incorporé à un module pour rendre son utilisation plus facile d'accès. On utilise la STM32F746NG Discovery pour tester le module.

Le capteur TOF nécessite L'I2C (Inter Integrated Circuit). Le bus I2C permet de faire communiquer entre eux des composants électroniques très divers grâce à trois fils.

- Un signal de donnée : **SDA**
- Un signal d'horloge : **SCL**
- Un signal de référence électrique : Masse

Nous alimentons le capteur en 3.3V, et le capteur possède également une broche XSHUT (une entrée) et la broche GPIO1 (une sortie à drain ouvert).

Le VL53L0X possède une librairie (un driver) complète, nécessaire pour pouvoir utiliser les fonctions associées (démarrage, lecture, …).
Le driver une fois téléchargé dans le projet, les fonctions sont simples à utiliser. L'I2C permet d'obtenir des valeurs rapidement, la conversion de ces dernières se fera via des fonctions de la librairie.

Une structure principale contient l'ensemble des éléments nécessaires à la liaison avec l'I2C.

```
typedef struct {
    VL53L0X_DevData_t Data;                 /*!
    /*!< user specific field */
    I2C_HandleTypeDef *I2cHandle;
    uint8_t   I2cDevAddr;
    char    DevLetter;
    int     Id;
    int     Present;
    int     Enabled;
    int     Ready;
    uint8_t   comms_type;
    uint16_t  comms_speed_khz;
    int LeakyRange;
    int LeakyFirst;
    uint8_t RangeStatus;
    uint8_t PreviousRangeStatus;
    FixPoint1616_t SignalRateRtnMegaCps;
    uint16_t EffectiveSpadRtnCount;
    uint32_t StartTime;
} VL53L0X_Dev_t;
```

Figure 11 Structure de la librairie du VL54L0X

Dans notre projet l'objectif est de pouvoir donner la distance entre le robot et une canette, de ce fait une fois que la distance sera égale, ou proche de zéro, les moteurs devront s'arrêter pour que le robot s'arrête et que la pince prenne le relais.

3.3.5 Servomoteur XL320

Le servomoteur va nous permettre d'attraper la canette. Dès que le robot va détecter une canette il va avancer vers celle-ci et dès que la distance est bonne, le servomoteur va entrer en action en ouvrant la pince pour que le robot puisse saisir la canette. Nous avons donc commencé par essayer de comprendre comment celui-ci pouvait fonctionner et comment on pouvait le faire tourner.

Figure 12 Servomoteur XL320

Mode de fonctionnement :

Le servomoteur nécessite l'envoie d'une trame. Différents modes sont disponibles, on teste dans un premier temps le mode « write » avec la gestion des LEDS et le mouvement du servomoteur.

Write Instruction Packet															
Header			Reserved	Packet ID	Packet Length		Instruction	Param1	Param2	Param3	Param4	Param5	Param6	16bit CRC	
								Address		Data					
0xFF	0xFF	0xFD	0x00	0x01	0x09	0x00	0x03	0x74	0x00	0x00	0x02	0x00	0x00	0xCA	0x89

Figure 13 Trame write à envoyer au servomoteur

Nous pouvons voir que les données Header, *reserved* sont fixés, nous n'avons pas à les modifier. Les bits *CRC* quant à eux sont calculés automatiquement dans le programme. Nous voulons l'utiliser en *broadcast* donc *le packet id est* fixé à la valeur donnée dans sa datasheet qui est de 0xFE. L'instruction correspond à ce que nous souhaitons faire. Pour écrire, on utilise la valeur 0x03.

Test LED :

Pour pouvoir allumer la led il faut écrire à l'adresse 0x19 qui correspond à la valeur 25 en décimal.

25	1	LED	LED On/Off

Figure 14 Commande de la LED

Nous avons testé de contrôler la LED que l'on peut mettre de différentes couleurs :

```
#define OFF 0x00
#define RED 0x01
#define GREEN 0x02
#define YELLOW 0x03
#define BLUE 0x04
#define PURPLE 0x05
#define CYAN 0x06
#define WHITE 0x07
```

Figure 15 Commandes couleurs de la LED du servomoteur

Test

On allume la LED et ensuite on l'éteint, donc il nous faut la valeur 0 pour éteindre et 2 pour avoir la LED en vert.

Après l'envoi du code, voici les trames obtenues :

```
ff ff fd 0 fe 6 0 3 19 0 2 d5 92
ff ff fd 0 fe 6 0 3 19 0 0 da 12
```

Figure 16 Trame reçue pour le fonctionnement de la LED

Les valeurs obtenues sont celles souhaitées.

Ensuite, nous sommes passé à tester la rotation du servo, pour cela la trame change car l'adresse change, maintenant nous avons l'adresse 0x1E pour le mode Goal Position :

30	2	Goal Position	Goal Position

Figure 17 Commande de la rotation du servomoteur

Nous entrons les paramètres souhaités avec la valeur que nous voulons, dans la datasheet nous avons trouvé que le servo-moteur pouvait aller de la position 0 à 1023 et en hexadécimal il faut écrire 0x00 pour la valeur 0 et 0x3FF.

Voici les trames obtenues :

```
ff ff fd 0 fe 7 0 3 1e 0 ff 3 79 35
ff ff fd 0 fe 7 0 3 1e 0 0 0 7c b7
```

Figure 18 Trame reçue pour la rotation du servomoteur

Si on regarde, nous pouvons voir que les premières données n'ont pas changé, nous changeons les données à partir de la valeur 7 car nous avons 7 valeurs par la suite avec l'instruction qui vaut 3 car on écrit, la valeur 1E correspond au mode que l'on veut donc pour nous la position, le ff 3 correspond à la position que l'on veut.

Dans notre cas nous testons pour la position 1023, juste qu'il faut faire attention car la première valeur correspond au bit de poids faible et les deux dernières valeurs sont calculées par la formule du code qui correspond au bit CRC.

La prochaine étape sera de gérer la vitesse du servomoteur et relier son fonctionnement, et donc l'ouverture de la pince lorsque l'on s'approchera d'une canette.

4 Conclusion

En 2 mois nous avons réalisé 90% de la partie hardware. Le PCB est produit, il ne manque que la soudure des composants.
Nous pourrons entamer les premiers tests sur la carte une fois que cette dernière sera prête avec l'ensemble des soudures réalisées.

Tous les capteurs ont un code qui est écrit/en cours d'écriture et testé sur la STM32F746NG Discovery. Les différents capteurs, le servomoteur et les moteurs devront agir ensemble, il nous faudra déterminer les différentes tâches à mettre en place (utilisation de FreeRTOS). Le diagramme d'états, une fois complet, nous permettra de définir les tâches à mettre en place.

Le design du robot a été repensé afin de pouvoir contenir l'ensemble des capteurs, une fois imprimé en 3D, une analyse sera effectuée quant aux emplacements des capteurs et le mode de fonctionnement de la pince, et une phase d'amélioration sera entamée pour obtenir un robot fonctionnel et où tous les capteurs auront un emplacement adapté (selon leur utilisation)

ANNEXES

Annexe 1 : schéma architectural

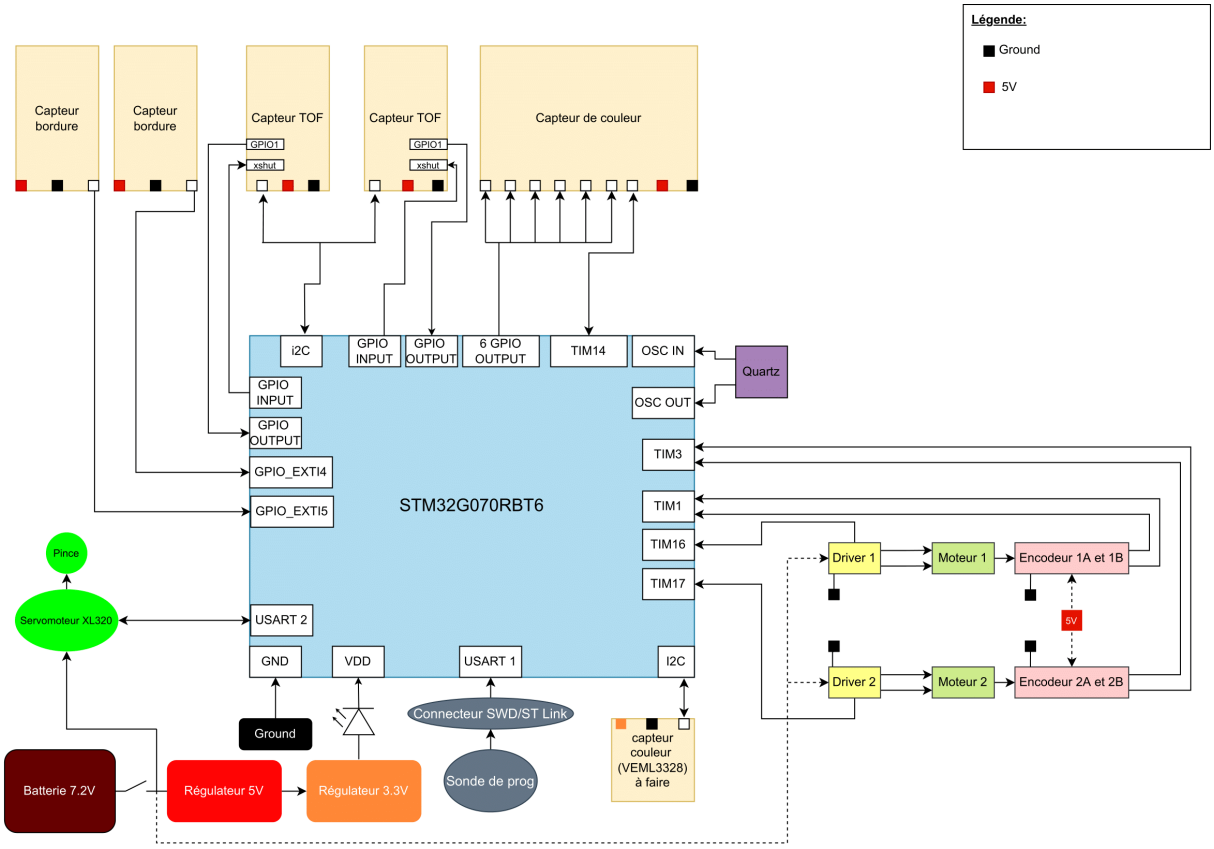


Figure 19 Schéma architectural

Annexe 2 : pinout view

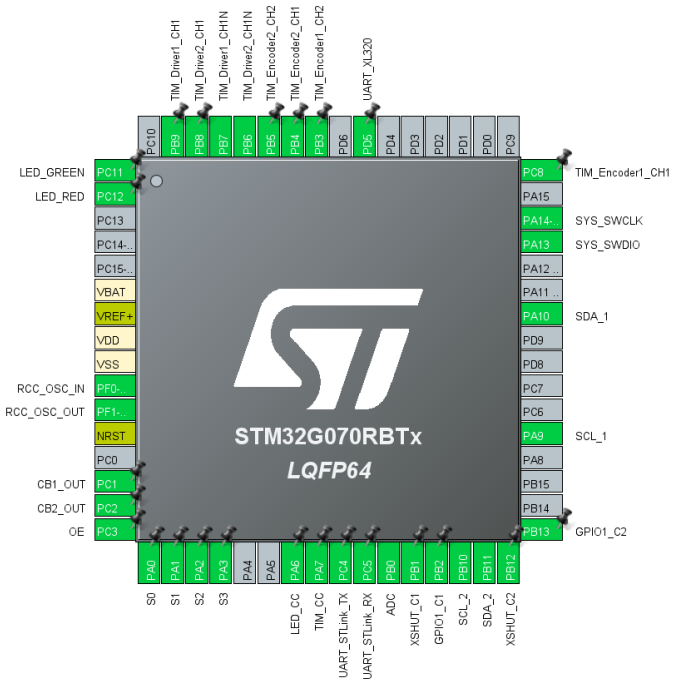
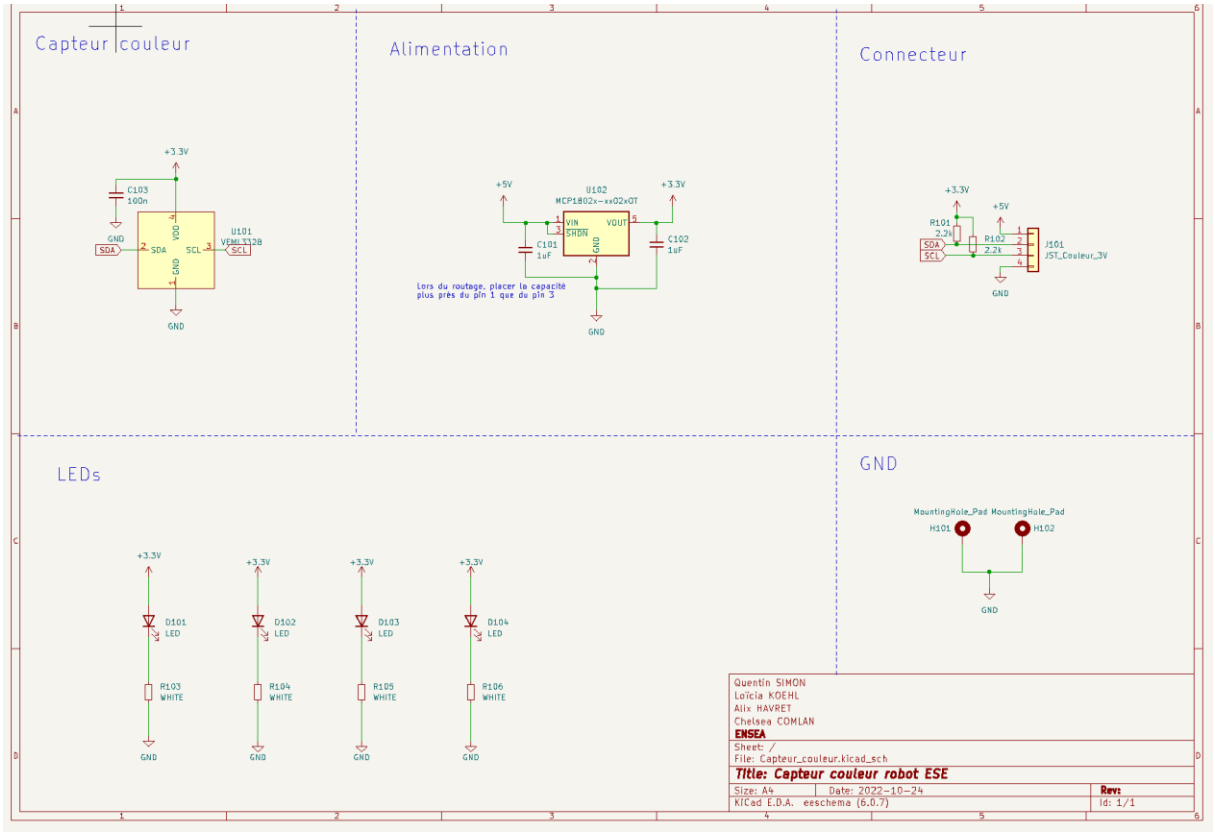


Figure 20 Pinout view sur CubeIDE

Annexe 3 : Schématique capteur couleur 3.3V



Annexe 4 : Liste des figures