

## Dossier De Conception (DDC)

du projet

# Robot Mini-Sumo

### Responsabilité documentaire

Action	NOM Prénom	Fonction	Date	Signature
Rédigé par	BINNER Antoine, BLANC Louis, PERON Nathan, LESTAGE Margot, CALLET Arnaud, LINDOIS Jérémy	Technicien	07/10/2024	
Approuvé par	<Chef du projet> (IUT GEII Bdx)	Chef de projet	JJ/MM/AAAA	
Approuvé par	<Client> (IUT GEII Bdx)	Client	JJ/MM/AAAA	

## Suivi des révisions documentaires

Indice	Date	Nature de la révision
1	01/09/2022	Publication préliminaire du DDC document à compléter par le Technicien.
2	07/10/2024	Première publication

## Documents de références

Sigle	Référence	Titre	Rév.	Origine
[CDC]	RMS_CDC	Cahier des charges	1	IUT GEii Bdx

## Table des matières

<b>1. Nature du document</b>	<b>5</b>
<b>2. Conception préliminaire du produit</b>	<b>5</b>
<b>2.1 Architecture Électronique</b>	<b>5</b>
2.1.1 Choix des capteurs adversaire	7
2.1.2 Choix des capteurs de sol	8
2.1.3 Choix du capteur infrarouge	12
2.1.4 Choix du pont en H	13
2.1.5 Choix du régulateur de tension	14
2.1.6 Choix de la batterie	15
2.1.7 Choix des LED	16
2.1.8 Choix partie gestion batterie	17
<b>2.2 Architecture Informatique</b>	<b>19</b>
2.2.1 Fonctions d'acquisition	20
2.2.2 Fonctions d'action	21
2.2.3 Fonctions de traitement	21
<b>2.3 Conclusion de la conception préliminaire du produit</b>	<b>27</b>
<b>3. Conception détaillée du produit</b>	<b>28</b>
<b>3.1 Conception détaillée du robot mini-sumo</b>	<b>28</b>
<b>3.2 Conception détaillée de la partie acquisition</b>	<b>28</b>
<b>3.2.1 Capteurs adversaires</b>	<b>29</b>
3.2.1.1 Schéma électrique des capteurs adversaires	29
3.2.1.2 Code informatique des capteurs adversaires	30
<b>3.2.2 Capteurs de sol</b>	<b>30</b>
3.2.2.1 Schéma électrique des capteurs de sol	30
3.2.2.2 Code informatique des capteurs de sol	34
<b>3.2.3 Capteurs de télécommande</b>	<b>34</b>
3.2.3.1 Schéma électrique des capteurs de télécommande	34
3.2.3.2 Code informatique des capteurs de télécommande	35
<b>3.3 Conception détaillée de la partie action</b>	<b>36</b>
<b>3.3.1 Pont en H</b>	<b>36</b>
3.3.1.1 Schéma électrique du pont en H	36
3.3.1.2 Code informatique du pont en H	37
<b>3.3.2 Diodes électroluminescentes</b>	<b>37</b>
3.3.2.1 Schéma électrique des LED	37
3.3.2.2 Code informatique de la LED bleue	38
<b>3.4 Conception détaillée de la partie énergie</b>	<b>38</b>
<b>3.4.1 Sécurisation de la batterie</b>	<b>38</b>

<b>3.4.2 régulateur de tension</b>	<b>40</b>
<b>3.4.2.1 Schéma électrique du régulateur de tension</b>	<b>40</b>
<b>3.5 Conception détaillée de la partie traitement</b>	<b>40</b>
<b>3.5.1 Le microcontrôleur ATMEGA328P sur carte de prototypage arduino uno</b>	<b>40</b>
<b>3.4.1.1 Schéma électrique du microcontrôleur</b>	<b>40</b>
<b>3.3.1.2 Code informatique du microcontrôleur</b>	<b>41</b>
<b>4. Dérisquage des solutions techniques retenues</b>	<b>44</b>
<b>4.1 Essai programme contrôle des moteurs</b>	<b>44</b>
<b>4.2 Essai programme LED</b>	<b>48</b>
<b>4.3 Essai programme Capteurs</b>	<b>49</b>
<b>4.4 Conclusion de la simulation / prototypage rapide du produit</b>	<b>53</b>
<b>5. Conclusion de la conception du produit</b>	<b>53</b>
<b>6. Matrice de conformité du produit</b>	<b>53</b>

## 1. Nature du document

Ce document est un dossier de conception et a pour but de détailler la conception du produit développé. Il apporte ainsi des preuves de la conformité du produit par rapport à l'ensemble des exigences client. Le paragraphe 3 du [CDC] décrit de façon plus détaillée la nature et le positionnement de ce document dans l'arborescence documentaire du projet.

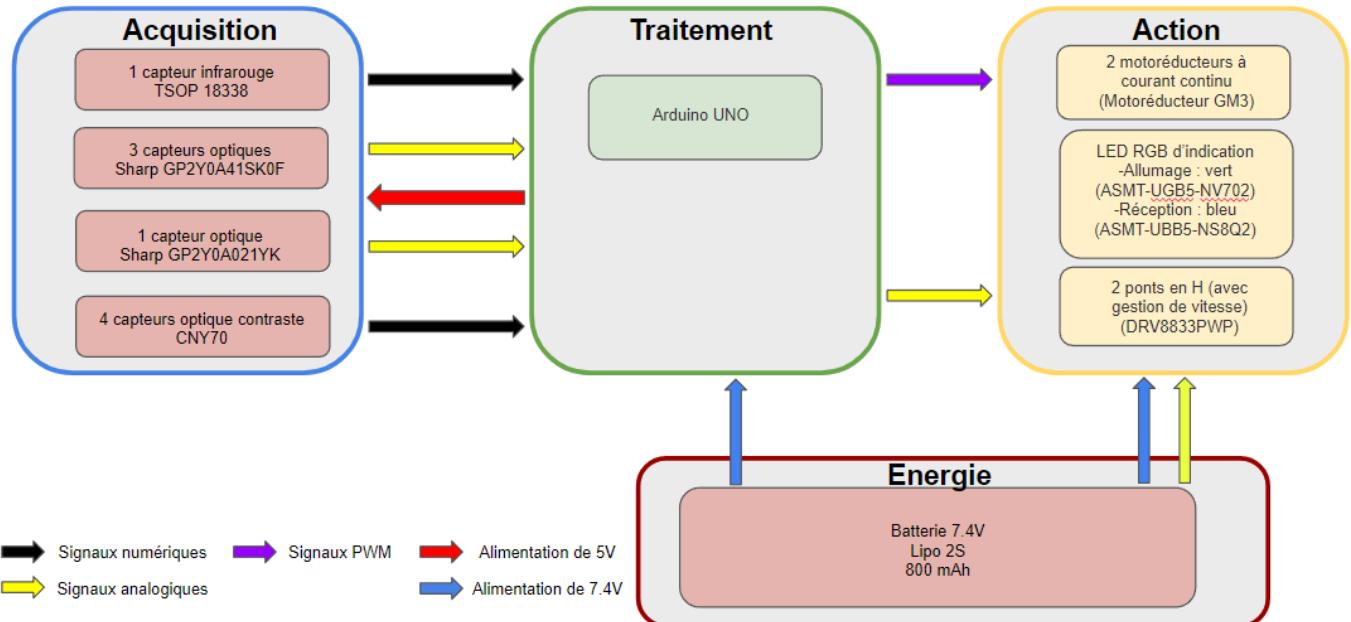
## 2. Conception préliminaire du produit

Ce chapitre décrit l'architecture fonctionnelle du produit. Il apporte les premiers éléments de preuve de la faisabilité du produit vis-à-vis des exigences client.

### 2.1 Architecture Électronique

**Référence de pré-conception :** CPR\_ARCHITECTURE

Afin de répondre au cahier des charges, une analyse globale des exigences a conduit à l'architecture fonctionnelle présentée ci-dessous.



**Figure 1 : Architecture électronique du robot sumo**

Premièrement, la partie acquisition permet de recueillir les informations concernant la position du robot ennemi et sa distance par rapport à notre robot, et la position de notre robot sur le terrain ("Dohyo").

La partie traitement comprend toutes les solutions techniques et informatiques permettant le traitement des informations acquises précédemment afin d'envoyer des instructions à la partie action. Une grosse partie informatique est incluse dans la partie traitement et permet la stratégie de combat générale de notre robot.

La partie action est la mise en œuvre des instructions données par la partie traitement. Elle permet de contrôler les moteurs et d'informer la mise sous tension du robot et la réception du signal de la télécommande grâce à deux diodes électroluminescentes.

Et finalement, la partie énergie permet d'alimenter tout le système grâce à une batterie Lipo 2S et un régulateur de tension.

### 2.1.1 Choix des capteurs adversaire

Référence de pré-conception : CPR\_CAPTEUR\_ADVERSAIRE

Exigences client vérifiées par préconception : EXIG\_ADVERSAIRE

FICHE D'AIDE A LA DECISION (FAD)								
Solution technique proposée	Conformité à toutes les exigences (oui : 1 / non : 0)	Critères de sélection (exemples) (valeur de 1 à 5)				Total (produit des valeurs précédentes)	Commentaires (si nécessaire)	
		Distance min et max de détection	Temps de réponse (en ms)	Faible consommation en courant (en mA)	Faible Prix (en €)			
Capteur de mesure Sharp GP2Y0A41SK0F	1	valeur sur datasheet	21.5 +/-3.7	12	8.25	4-30	400	
		score /5	4	4	5	5		
Capteur de mesure Sharp GP2Y0A021YK	1	valeur sur datasheet	43.3 +/-9.6	30	9.58	10-80	80	
		score /5	2	2	4	5		
Capteur de mesure Sharp GP2Y0A02YK	1	valeur sur datasheet	38.3 +/-9.6	33	10.67	20-150	45	
		score /5	3	1	3	5		
IUT de Bordeaux Département GEII	Référence : SUMO_FAD Révision : 1.6 – 05/10/2024							3 / 7

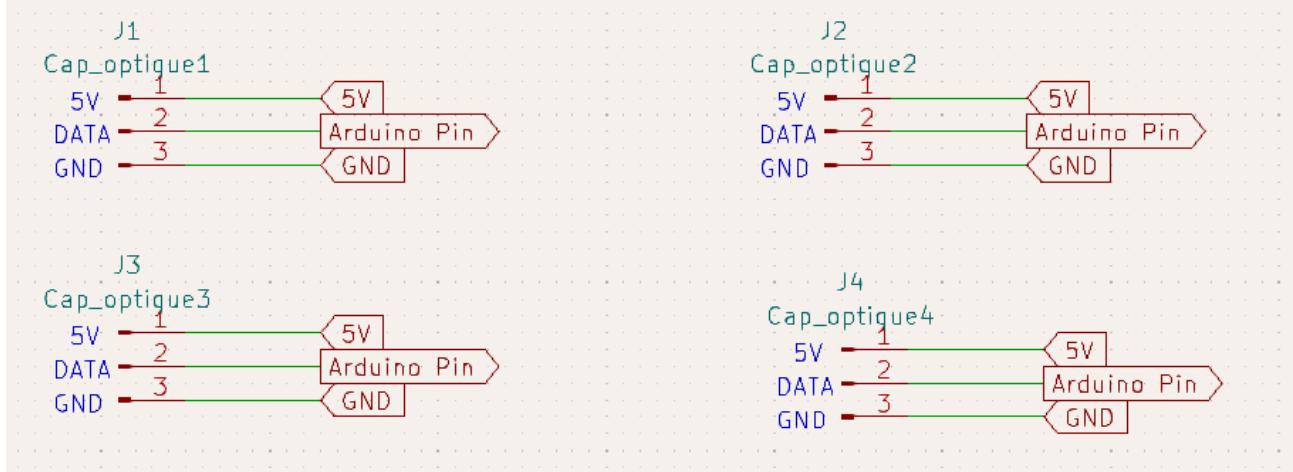
Figure 2 : FAD pour le choix des capteurs optiques

Nous avons élaboré une stratégie de combat axée sur la détection précise de la position de l'adversaire. Pour cela, nous avons opté pour un système de quatre capteurs optiques positionnés stratégiquement sur notre robot. Trois d'entre eux, placés à l'avant et sur le côté gauche, seront des capteurs à courte portée, tandis que le dernier, à droite, aura une portée plus étendue. Cette configuration nous permettra de localiser l'adversaire en effectuant une rotation sur place, éliminant ainsi la nécessité de nous déplacer à l'aveugle.

Pour répondre à cela nous utiliserons 3 capteurs Sharp GP2Y0A41SK0F qui correspondent à nos attentes et qui se distinguent notamment par leurs temps de réponse. Quant au capteur longue portée, nous avons opté pour le Sharp GP2Y0A021YK, capable de fournir des mesures fiables de 5 cm à 80 cm, ce qui correspond parfaitement à notre stratégie de détection et de combat. Ces capteurs garantissent une localisation précise de l'adversaire tout en optimisant le temps de réaction de notre robot.

## Robot Mini-Sumo (RMS)

Notre schéma électrique préliminaire pour acquérir les valeurs des 4 capteurs optiques est ci-contre :



**Figure 3 : Schéma électrique préliminaire des capteurs**

### 2.1.2 Choix des capteurs de sol

Référence de pré-conception : CPR\_CAPTEUR\_SOL

Exigences client vérifiées par préconception : EXIG\_LUMINOSITE

FICHE D'AIDE A LA DECISION (FAD)							
Solution technique proposée	Conformité à toutes les exigences (oui : 1 / non : 0)	Critères de sélection (exemples) (valeur de 1 à 5)				Total (produit des valeurs précédentes)	Commentaires (si nécessaire)
		Distance min et max de détection	Tension d'alimentation (en V)	Respect de la stratégie de combat	Faible Prix (en €)		
		valeur sur datasheet	5	x	1,39		
2 CNY avant + 2 CNY arrière	1	score /5	5	5	5	125	Résistances préalablement soudé
IUT de Bordeaux Département GEII	Référence : SUMO_FAD Révision : 1.6 – 05/10/2024						4 / 7

**Figure 4 : FAD pour le choix du nombres de capteurs de contraste**

Dû à notre stratégie de combat de notre robot sumo, nous avons choisi d'utiliser 4 capteurs de contraste, 2 à l'avant et 2 à l'arrière, placés sous le robot.

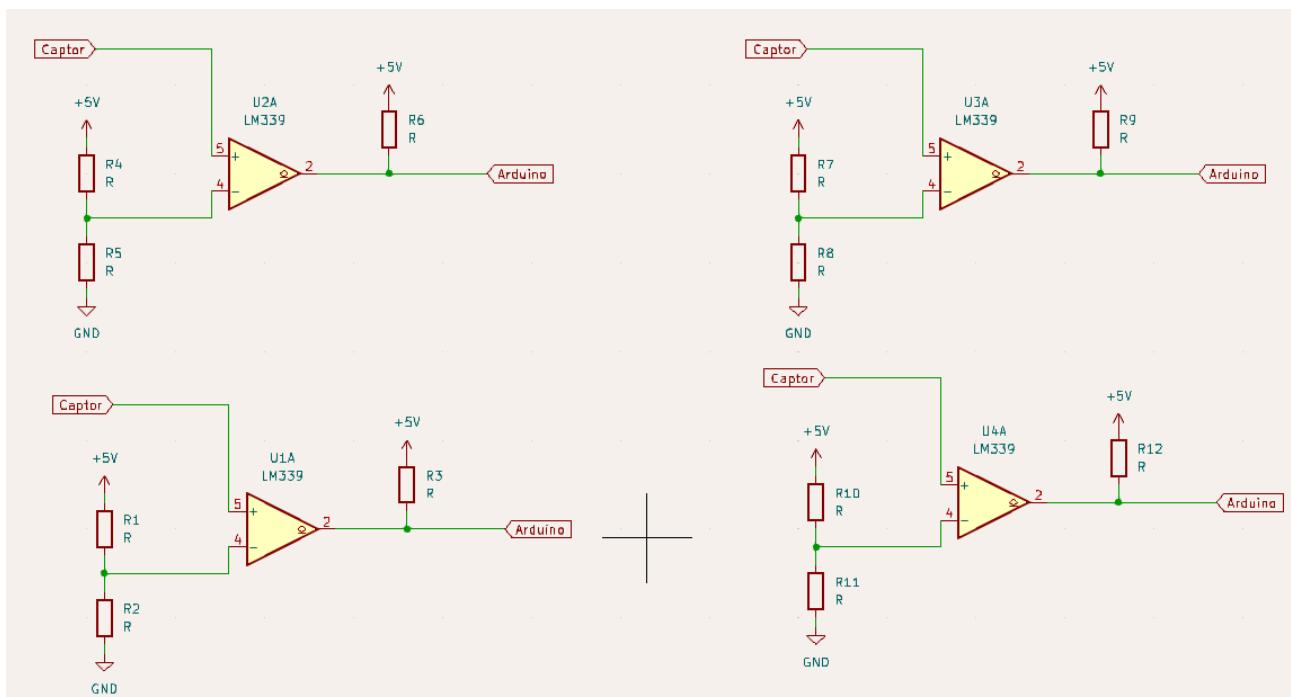
Cependant, ce choix nous confronte à un problème car nous aurons aussi 4 capteurs optiques et 1 capteur infrarouge qui renvoient tous deux des valeurs analogiques, comme le capteur de contraste. La carte n'ayant que 6 broches analogiques, nous ne pouvons pas brancher directement tous les capteurs analogiques à la carte électronique.

Nous avons donc pris le choix d'utiliser des comparateurs pour transformer la valeur analogique renvoyée par les capteurs de contraste en valeur binaire afin de n'utiliser que 2 entrées analogiques.

Tout d'abord, nous devons réaliser un pont diviseur de tension pour réaliser la tension de comparaison. Pour connaître cette tension, nous avons réalisé le test ci-dessous à l'aide d'une carte Arduino et d'une breadboard pour mesurer la tension lorsque le capteur de contraste mesure une couleur noire et une couleur blanche.

Nous rajouterons une résistance de pull-up à la sortie de notre comparateur pour éviter d'avoir une tension flottante.

Notre schéma électrique préliminaire pour acquérir les valeurs des 4 capteurs de contraste est ci-contre :



**Figure 5 : Schéma électrique préliminaire des capteurs de contraste**

De plus, pour répondre à l'exigence de luminosité nous avons procédé à deux tests.

Le premier est pour tester les résultats du capteur si la luminosité est basse pour simuler une pièce plonger dans le noir, pour ce faire nous avons placé une boîte opaque au dessus de notre capteur.

## Robot Mini-Sumo (RMS)

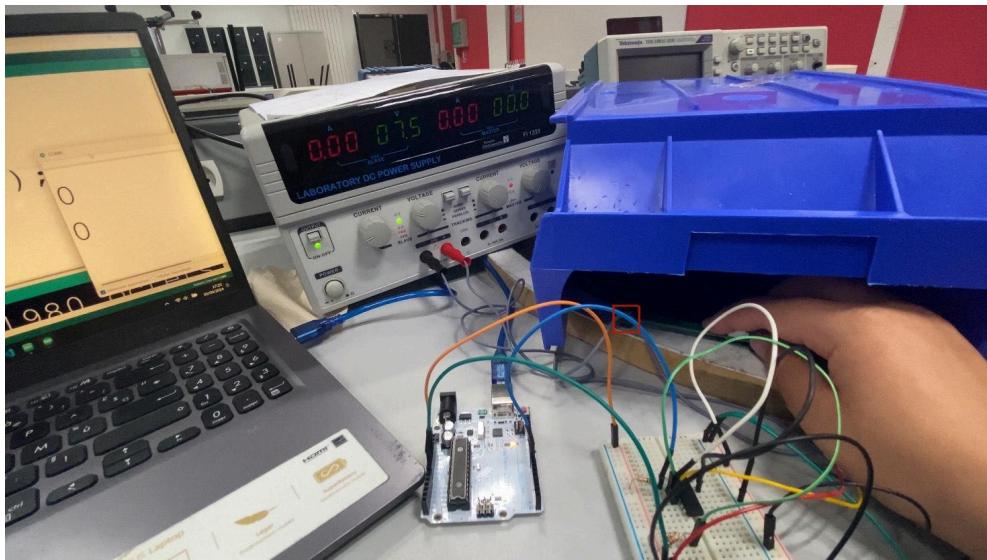


Figure 6 : Test luminosité basse sur ligne blanche

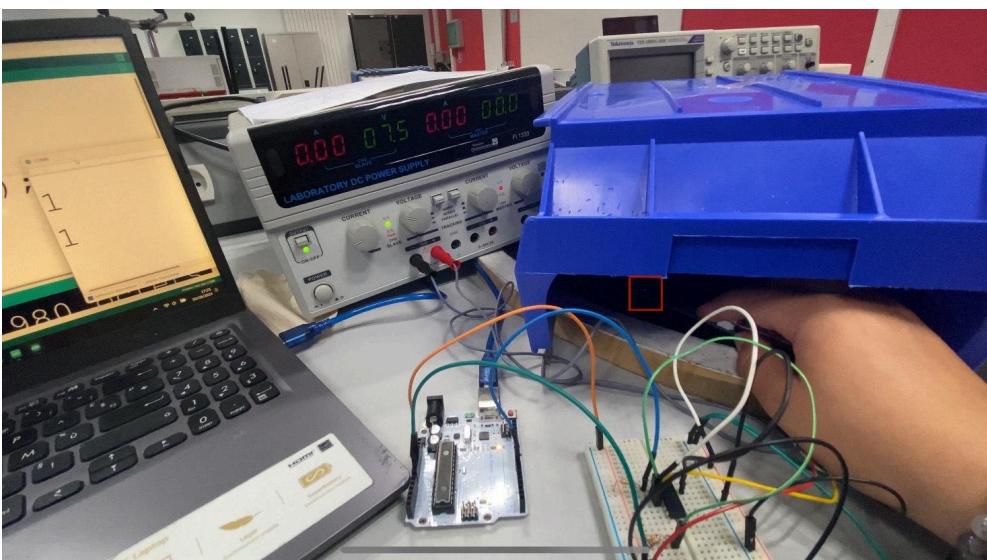
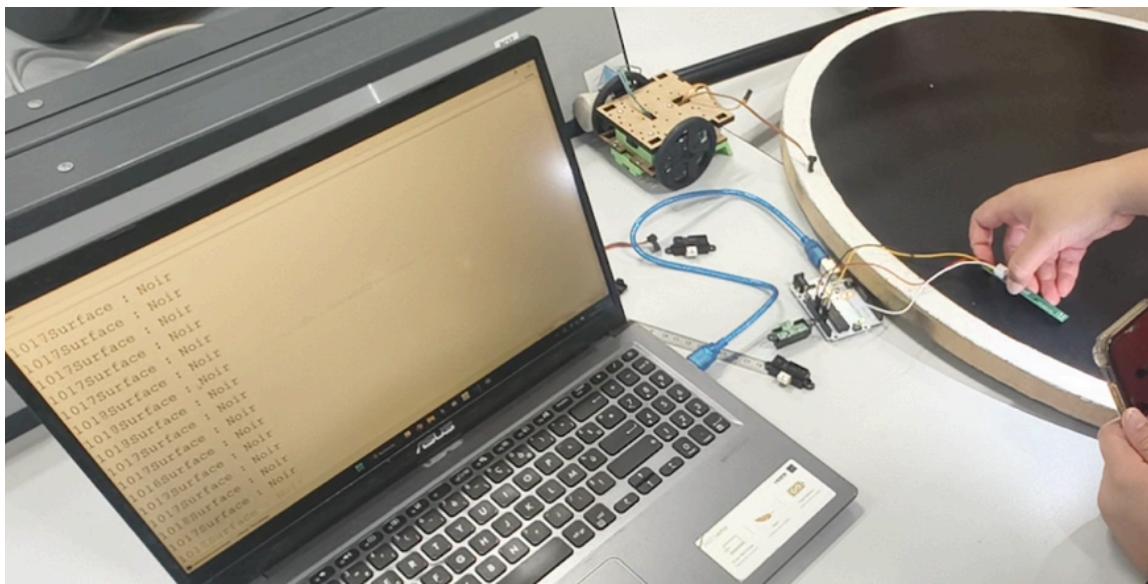


Figure 7 : Test luminosité basse sur couleur noire

Pour le second test nous avons utilisé le flash de nos téléphones pour simuler des possibles flash d'appareils photos.

## Robot Mini-Sumo (RMS)



**Figure 8 : Test luminosité haute sur couleur noire**



**Figure 9 : Test luminosité haute sur ligne blanche**

Grâce à nos tests, nous en concluons que les capteurs répondent à l'exigence luminosité soumis par le client.

### 2.1.3 Choix du capteur infrarouge

Référence de pré-conception : CPR\_CAPTEUR\_INFRAROUGE

Exigences client vérifiées par préconception : EXIG\_DEPART

FICHE D'AIDE A LA DECISION (FAD)						
Solution technique proposée	Conformité à toutes les exigences (oui : 1 / non : 0)	Critères de sélection (exemples) (valeur de 1 à 5)			Total (produit des valeurs précédentes)	Commentaires (si nécessaire)
		Distance min et max de détection	Tension d'alimentation (en V)	Longueur d'onde recommandé (en nm)		
TSOP 18338	1	valeur sur datasheet score /5	2,5 - 5,5 5	950 5	1,39 5	125
IUT de Bordeaux Département GEII	Référence : SUMO_FAD Révision : 1.6 – 05/10/2024					4 / 7

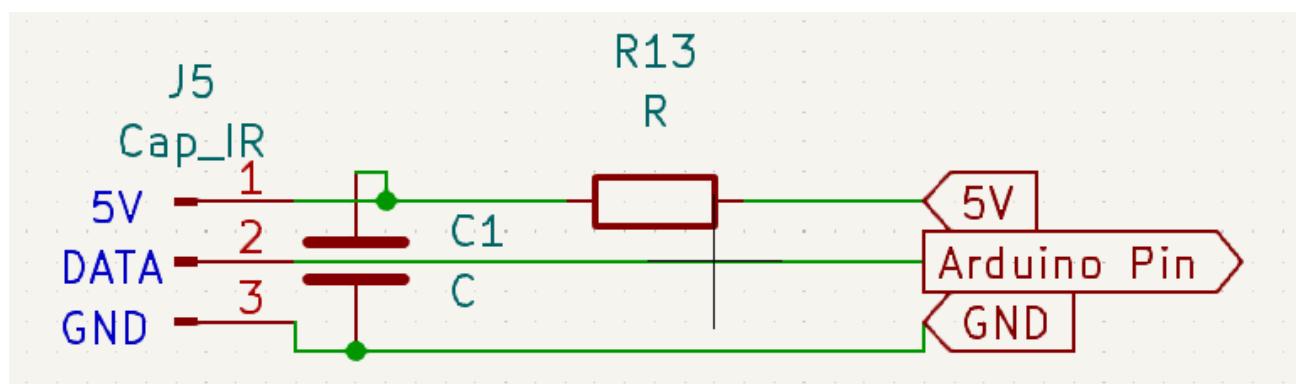
Figure 10 : FAD pour le choix du capteur infrarouge

Nous avons choisi ce capteur car il répond à tous les besoins et exigences et que c'est le seul disponible en stock lors de la rédaction de ce dossier de conception.

Le but de ce capteur sera de savoir si nous aurons reçu une information infrarouge ou non.

Pour pouvoir réaliser la programmation de ce capteur, nous utiliserons la librairie Arduino *IRremote*.

Notre schéma électrique préliminaire pour acquérir les valeurs des 4 capteurs optiques est ci-contre :



**Figure 11 : Schéma électrique préliminaire du capteur infrarouge**

A l'aide de la page numéro deux de la datasheet du capteur infrarouge, nous savons que nous devons ajouter un condensateur et une résistance comme le schéma ci-dessus.

#### 2.1.4 Choix du pont en H

Référence de pré-conception : CPR\_PONT\_EN\_H

Exigences client vérifiées par préconception : EXIG\_DEPLACEMENT

Grâce à notre fiche d'aide à la décision ci-dessous (voir figure 12), nous avons choisi le pont en H DRV8833.

Solution technique proposée	Conformité à toutes les exigences (oui : 1 / non : 0)	Critères de sélection (exemples) (valeur de 1 à 5)					Commentaires (si nécessaire)
		Compatibilité de la tension avec les sources disponibles	Courant max de sortie du pont	Manière de piloter le pont	Faible Prix	Disponibilité	
DRV8835	1	0V-11V	1.5A	PWM 2V-7V	4.99€	1	Dû à des difficultés de fabrication, nous ne choisissons pas cette solution technique
		5	4	5	4	5	
DRV8833	1	2.7V-10.8V	2A	PWM 3V-5V	9.50€	1	Dû à des difficultés de fabrication, nous ne choisissons pas cette solution technique
		3	4	4	2	5	
DRV8830	0	2.75V - 6.8V	1A	BUS I2C	12.75€	0	0
		2	0	0	1	0	
DRV8801	0	8V - 38V	2.8A	PWM 3.3V-6.5V	11.65€	0	0
		1	5	4	1	0	

**Figure 12 : FAD pour le choix du pont en H**

Tout d'abord nous avions le choix entre quatre ponts en H. Après analyse des datasheets seulement deux étaient conformes au cahier des charges : le DRV8835 et le DRV8833. Ensuite, nous avons comparé les différentes caractéristiques des ponts en H et avons obtenu des meilleures spécifications sur le DRV8835. Cependant, nous n'avions besoin que de la puce dans le driver comprenant le pont en H, et la puce du DRV8835 est trop difficile à souder à notre niveau et avec notre matériel. Nous avons donc choisi le DRV8833.

### 2.1.5 Choix du régulateur de tension

Référence de pré-conception : CPR\_REG

Exigences client vérifiées par préconception : Sans objet

Grâce à notre FAD (figure 13), nous avons pu déterminer le choix du régulateur linéaire. Nous avons choisi le NCP1117ST50T3G.

Solution technique proposée	Conformité à toutes les exigences (oui : 1 / non : 0)	Critères de sélection (exemples) (valeur de 1 à 5)					Boîtier composant facile à utiliser/soudé	Total (produit des valeurs précédentes)	Commentaires (si nécessaire)
		Tension max d'entrée	Tension de sortie	Faible chute de tension sortie-entrée	Courant de sortie max				
LP2985AIM5-3.3/NOPB 3.3V CMS	0	16V	3.3V	300mV	150mA	SOT-23	0	0	
		5	3	5	0	5			
LP2985AIM5-5.0/NOPB 5V CMS	0	16V	5V	300mV	150mA	SOT-23	0	0	
		5	5	5	0	5			
MC33269DT-3.3G IC	1	20V	3.3V	1V	800mA	TO-220	1500	1500	
		5	3	4	5	5			
MC33269DT-5.0G IC	1	20V	5V	1V	800mA	DPAK	2500	2500	
		5	5	4	5	5			
NCP1117ST50T3G	1	12V	5V	1.07	800mA	SOT-223	2500	2500	
		5	5	4	5	5			
IUT de Bordeaux Département GEII	Référence : SUMO_FAD Révision : 1.6 – 05/10/2024							6 / 7	

Figure 13 : FAD pour le choix du régulateur de tension

Pour choisir ce régulateur linéaire, nous avons consulté sa datasheet en regardant la plage de tension qu'il pouvait recevoir, qui est de 6,5V à 12V (page 3 de la datasheet). Nous avons ensuite vérifié la tension de sortie, qui est de 5V +/- 0,1V (page 3 de la datasheet). Nous avons également examiné la tension de chute (qui permet de savoir quand le régulateur linéaire peut fournir la tension de sortie par rapport à l'entrée) de 1,07V dans le cas d'une sortie de 5V (page 4 de la datasheet). Par la suite, nous avons vérifié le courant maximum qu'il pouvait fournir, qui est de 800mA (page 3 de la datasheet). Pour finir, nous avons regardé le boîtier, qui est le SOT-223 (page 13 de la datasheet). Cela nous a donné deux composants possibles, mais nous avons choisi le NCP1117ST50T3G car c'est le régulateur linéaire de l'Arduino, ce qui nous évitera d'ajouter des composants supplémentaires et réduira le coût du composant.

### 2.1.6 Choix de la batterie

**Référence de pré-conception :** CPR\_BAT

**Exigences client vérifiées par préconception :** EXIG\_AUTONOMIE

Grâce à notre FAD (figure X), nous avons pu déterminer le choix de la batterie. Nous avons choisi la Lipo CONRAD ENERGY 7.4V 800mAh.

Solution technique proposée	Conformité à toutes les exigences (oui : 1 / non : 0)	Critères de sélection (exemples) (valeur de 1 à 5)			Total (produit des valeurs précédentes)	Commentaires (si nécessaire)
		Encombrement	Fab. Prix	autonomie		
Lipo CONRAD ENERGY 7.4 V 1200 mAh	0	72*36*12mm 0	17,99 3	1200 5	0	
Lipo CONRAD ENERGY 7.4 V 800 mAh	1	56*32*17mm 5	12,99 4	800 5	100	
Lipo CONRAD ENERGY 7.4 V 450 mAh	1	56*32*11,5 5	8,99 5	450 4	100	
IUT de Bordeaux	Référence : SUMO_FAD					
Département GEII	Révision : 1.6 – 05/10/2024				7 / 7	

**Figure 14 : FAD pour le choix de la batterie**

Nous avons choisi la batterie de 800mAh car elle avait des dimensions de 56\*32\*17 (page 3 de la datasheet), ce qui répond à notre exigence de dimension. Ensuite, nous avons regardé son prix, qui est de 12,99€ (page 1 de la datasheet).

Nous avons également examiné son autonomie, qui est de 800mAh (page 3 de la datasheet). Pour cette autonomie (appelée Q ou capacité), nous avons vérifié si elle correspondait à notre besoin. Nous avons dû estimer la consommation de chacun des composants, pour le moteur, nous avons effectué un test qui nous a donné 89,3 mA. Pour les deux capteurs de distance (GP2Y...) et les capteurs de contraste, nous avons demandé à nos coéquipiers s'occupant de la partie acquisition et qui travaillaient donc sur les capteurs, ce qui nous a donné 30mA, 12mA et 20mA.

Pour le driver, nous avons demandé à nos coéquipiers s'occupant de la partie action, qui nous ont indiqué 1,7mA. Pour le capteur IR, nous avons trouvé 3mA, et enfin pour l'Arduino, nous avons trouvé 9,5mA (page 596 de la datasheet). Après avoir collecté toutes ces valeurs, nous les avons compilées dans un tableau (figure 15) qui nous permet de visualiser la consommation totale de notre robot.

Composant	Consommation	Nombre	Consommation par composant
Moteur	89,3	2	178,6
GP2Y0A21YK	30	2	60
GP2Y0A41SK0F	12	2	24
Capteur de contraste	20	4	80
Driver	1,7	1	1,7
Capteur IR	3	1	3
Arduino	9,5	1	9,5
<b>Consommation Totale</b>	<b>356,8</b>		
<b>Consommation sans moteur</b>	<b>178,2</b>		

**Figure 15 : FAD pour le choix de la batterie**

D'après notre tableau, nous voyons que nous consommons 356.8mA puis d'après l'exigence autonomie du cahier des charges, le robot doit fonctionner pendant 90 minutes minimum ce qui nous donne  $t = 1.5h$  donc  $Q_{robot} = 356.8 * 1.5 = 568.2\text{mAh}$ . puis notre batterie fait 800mAh mais nous ne devons pas la décharger en dessous de 20% donc  $Q_{bat} = 800 * 0.8 = 640\text{mAh}$  donc notre robot a une autonomie de 1.69h ( 1h 42minutes )

### 2.1.7 Choix des LED

Référence de pré-conception : CPR\_LED

Exigences client vérifiées par préconception : Sans objet

Dans le but d'informer l'utilisateur sur l'état du robot, nous avons décidé d'installer deux diodes électroluminescentes (LED) de différentes couleurs :

- une verte qui s'allume lorsque le robot sumo est sous tension
- une bleue qui s'éclaire et clignote à intervalle régulier lors de la réception des ordres envoyés par la télécommande.

Nous devrons donc installer une résistance en série pour chacune des LED dans le but de gérer le courant les traversant et ainsi contrôler la luminosité de celles-ci.

La LED verte sera rattachée à la partie énergie du robot tandis que la LED bleue sera reliée à une sortie de notre microcontrôleur.

### 2.1.8 Choix partie gestion batterie

Référence de pré-conception : CPR\_GESTION\_BAT

Exigences client vérifiées par préconception : EXIG\_SECUR\_BATT

Pour protéger la batterie nous avons fait le choix de passer par un système électrique.

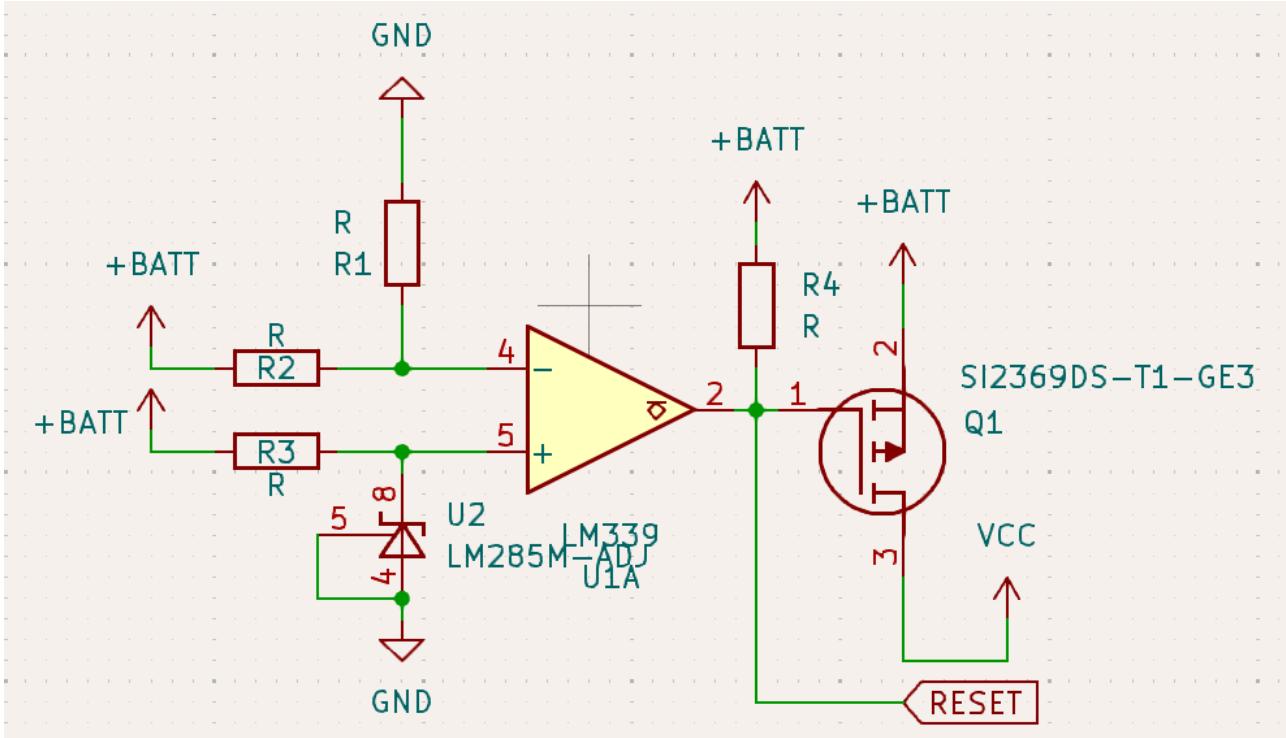


Figure 16 : schéma électrique du système de coupure des moteurs

Dans ce montage nous avons plusieurs parties :

- Référence de tension pour la comparaison [composant R3 et U2]  
Le composant U2 permet de récupérer une tension stable de 2.5V à ses bornes, cela nous sert de tension de référence pour le comparateur
- Tension batterie descendue [composant R2 et R1]  
Le pont diviseur de tension formé par ces deux résistances nous permet de descendre la tension, lorsque la batterie sera à 6.7V le pont diviseur de tension sortira une tension de 2.5V
- Le comparateur [composant U1A]  
Lorsque la tension de référence est plus petite que la tension de la batterie, alors le

comparateur met sa sortie à la masse, dans le cas contraire il ne met pas sa sortie à la masse (il ne l'a pas mis non plus à la tension d'alimentation car le comparateur est en collecteur ouvert).

- Le pull-up en sortie du comparateur [composant R4]

Lorsque la sortie du comparateur n'est pas mise à la masse alors la résistance R4 permet de mettre la sortie à la tension de la batterie.

- Le transistor de coupure de l'alimentation des moteurs [composant Q1]

Lorsque la tension sur la base du transistor est  $V_{bat}$  alors alors le circuit est coupé

Lorsque la tension sur la base du transistor est 0 alors le transistor est passant et  $V_{cc} = V_{bat}$

## 2.2 Architecture Informatique

Référence de pré-conception : CPR\_ARCH\_INFO

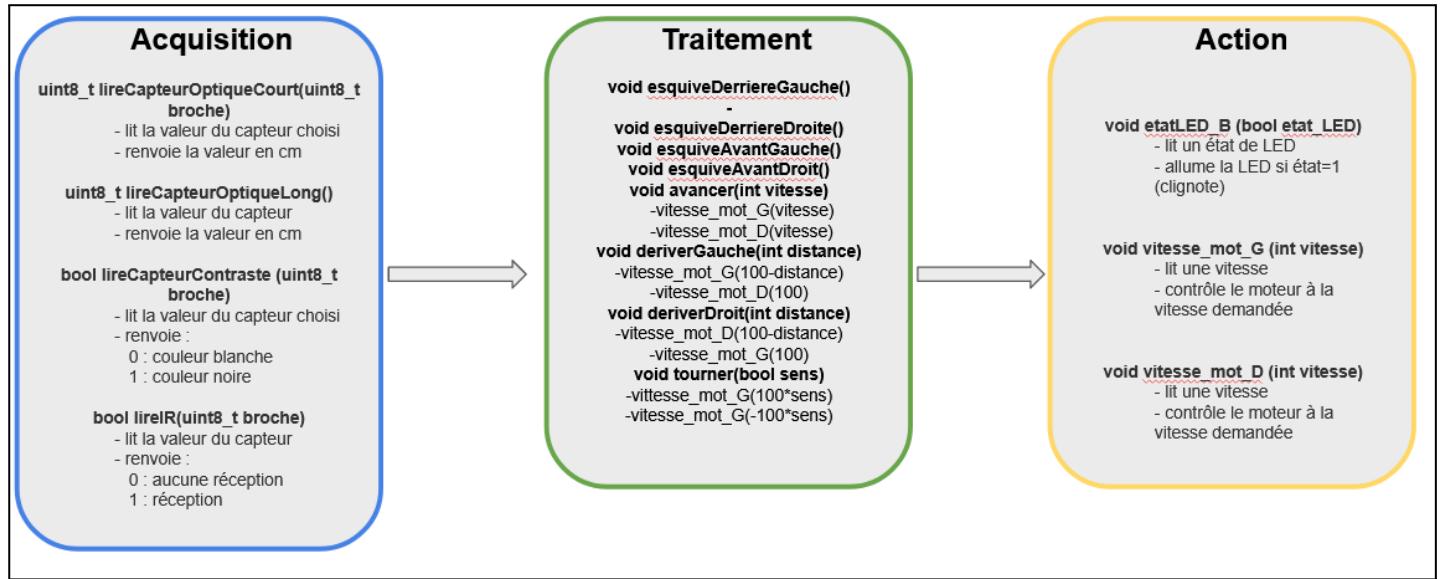


Figure 17 : Schéma de l'architecture informatique du robot sumo

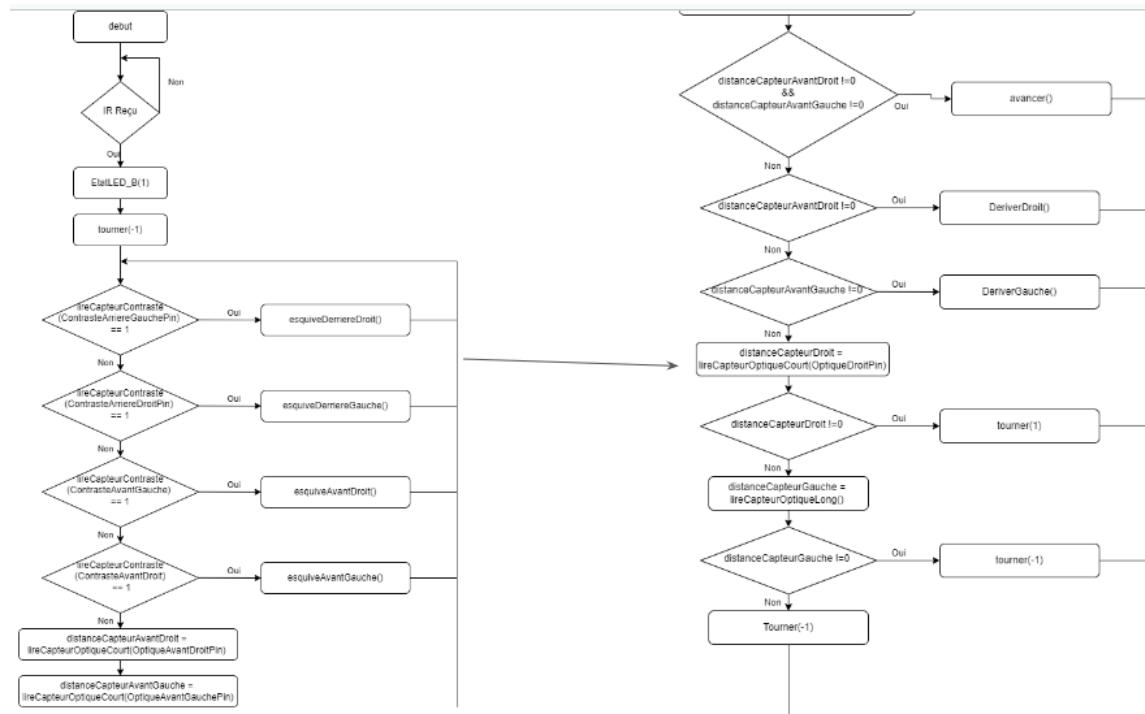


Figure 18 : Algorigramme du robot sumo

### 2.2.1 Fonctions d'acquisition

**Référence de pré-conception :** CPR\_ACQUI\_INFO

**Exigences client vérifiées par préconception :** EXIG\_ADVERSAIRE, EXIG\_DEPART

*Explications des différentes fonctions d'acquisition présentées dans le diagramme ci-dessus :*

#### **uint8\_t lireCapteurOptiqueCourt(uint8\_t broche)**

Cette fonction aura comme but de renvoyer la distance en cm entre le robot sumo et l'obstacle le plus proche mesuré par les capteurs dit "court".

La fonction renvoie une valeur entière positive sous un octet. Elles prendront comme paramètre la broche analogique du capteur optique choisie.

#### **uint8\_t lireCapteurOptiqueLong()**

Cette fonction aura comme but de renvoyer la distance en cm entre le robot sumo et l'obstacle le plus proche mesuré par le seul capteur dit "long"

La fonction renvoie une valeur entière positive sous un octet.

#### **bool lireCapteurContraste(uint8\_t broche)**

Cette fonction aura comme but de renvoyer une valeur binaire, soit 0 ou 1, si le sol est respectivement de couleur blanche ou noire.

Elle prendra comme paramètre la broche du capteur de contraste choisie.

#### **bool lireIR(uint8\_t broche)**

Cette fonction aura comme but de renvoyer une valeur binaire si le capteur infrarouge reçoit une quelconque information infrarouge ou non.

Elle prendra comme paramètre la broche du capteur infrarouge.

## 2.2.2 Fonctions d'action

**Référence de pré-conception :** CPR\_ACTION\_INFO

**Exigences client vérifiées par préconception :** EXIG\_DEPLACEMENT

Pour le déplacement du robot et donc le fonctionnement des moteurs, nous utilisons deux fonctions informatiques : une pour le moteur de gauche et une pour le moteur de droite.

Nous avons appelé ces fonctions **vitesse\_mot\_G** pour le contrôle du moteur gauche et **vitesse\_mot\_D** pour le contrôle du moteur de droite. Elles prennent en entrée chacune une variable comprise entre -100 et 100, -100 étant la vitesse maximale vers le sens arrière du robot et 100 la vitesse maximale vers l'avant.

Nous réaliserons donc un produit en croix pour déterminer la valeur à envoyer au moteur sur la broche de contrôle de vitesse envoyant un signal PWM compris entre 0 et 255.

Nous avons aussi créé une fonction pour l'allumage de la LED permettant l'information de la réception du signal de la télécommande. La fonction se nomme **etatLED\_B** et prend en entrée l'information envoyée par la partie traitement, soit son état allumé ou éteint.

## 2.2.3 Fonctions de traitement

**Référence de pré-conception :** CPR\_TRAITEMENT\_INFO

**Exigences client vérifiées par préconception :** EXIG\_COMPORTEMENT

*Explication des fonctions :*

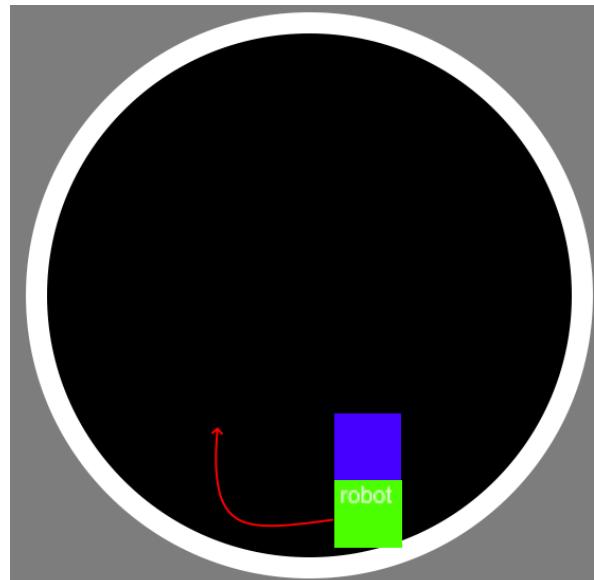
Carré vert = Robot sumo

Carré bleu = Robot adversaire

Flèches rouges = direction du robot sumo

- **void esquiveDerriereGauche()**

cet fonction reçoit aucun paramètre et envoyé aucun paramètre

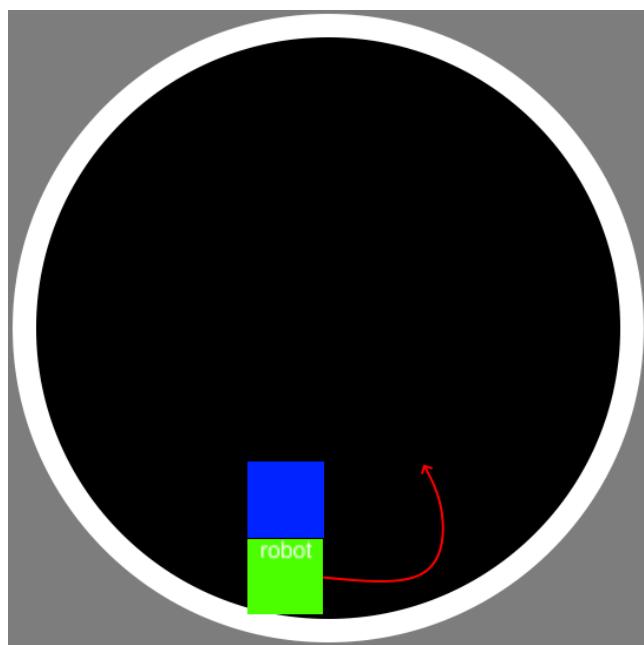


**Figure 19 : déplacement de derrière vers la gauche**

Sur cette figure nous voyons que notre robot va essayer d'esquiver le robot adverse en se déplaçant vers l'arrière gauche. Pour faire ce mouvement ont utilisé le moteur de gauche à 50% et le moteur de droite à 100%

- **void esquiveDerriereDroite()**

cet fonction reçoit aucun paramètre et envoyé aucun paramètre

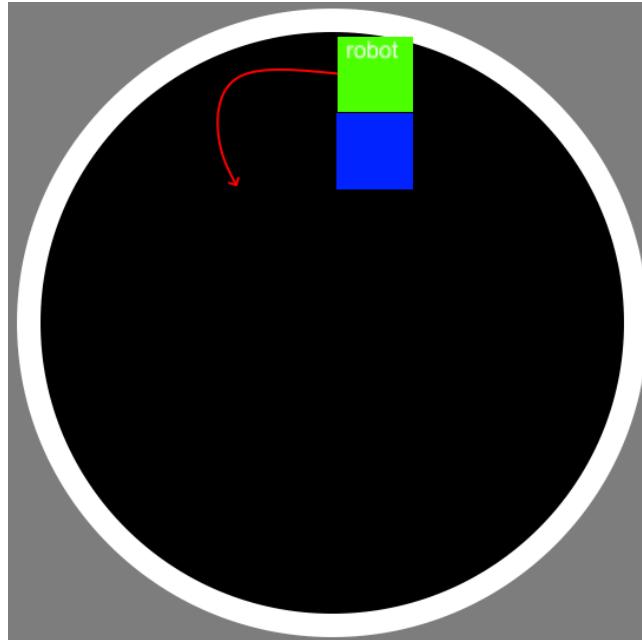


**Figure 20 : déplacement de derrière vers la droit**

Sur cette figure nous voyons que notre robot va essayer d'esquiver le robot adversaire en se décalant vers l'arrière droite. Pour faire ce mouvement ont utilisé le moteur de gauche à 100% et le moteur de droite à 50%

- **void esquiveAvantGauche()**

cet fonction reçoit aucun paramètre et envoyé aucun paramètre

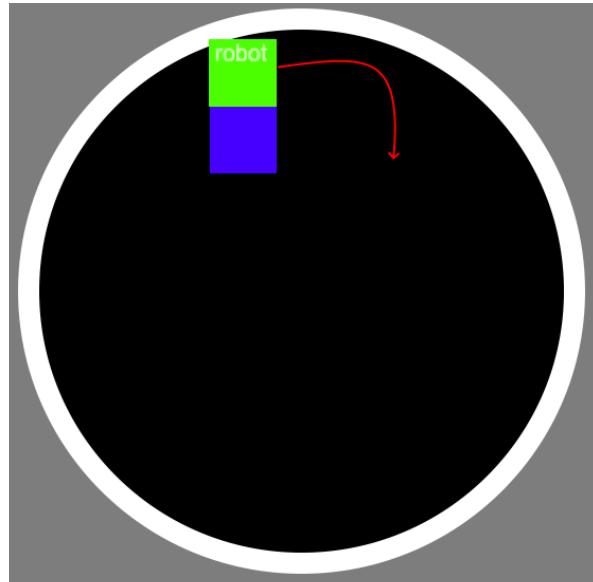


**Figure 21 : déplacement de avant vers la gauche**

Sur cette figure nous voyons que notre robot va essayer d'esquiver le robot adversaire en se décalant vers l'avant gauche. Pour faire ce mouvement ont utilisé le moteur de gauche à 50% et le moteur de droite à 100%

- **void esquiveAvantDroit()**

cet fonction reçoit aucun paramètre et envoyé aucun paramètre



**Figure 22 : déplacement de avant vers la droit**

Sur cette figure nous voyons que notre robot va essayer d'esquiver le robot adversaire en se décalant vers l'avant droite. Pour faire ce mouvement ont utilisé le moteur de gauche à 100% et le moteur de droite à 50%

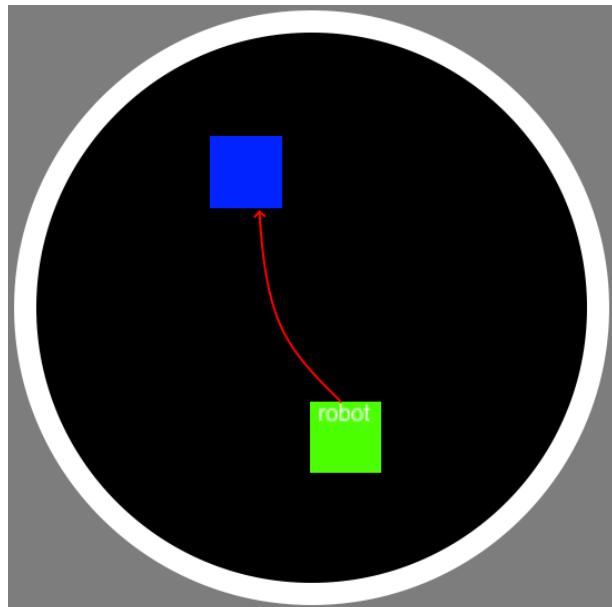
- **void avancer(vitesse)**

cet fonction reçoit un paramètre qui est la vitesse des moteur et envoyé aucun paramètre

Sur cette fonction nous allons faire avancer notre robot sumo vers le robot adversaire. Pour faire ce mouvement ont utilisé le moteur de gauche à 100% et le moteur de droite à 100%

- **void deriverGauche(int distance)**

cet fonction reçoit un paramètre qui est la distance que le capteur voix et envoyé aucun paramètre



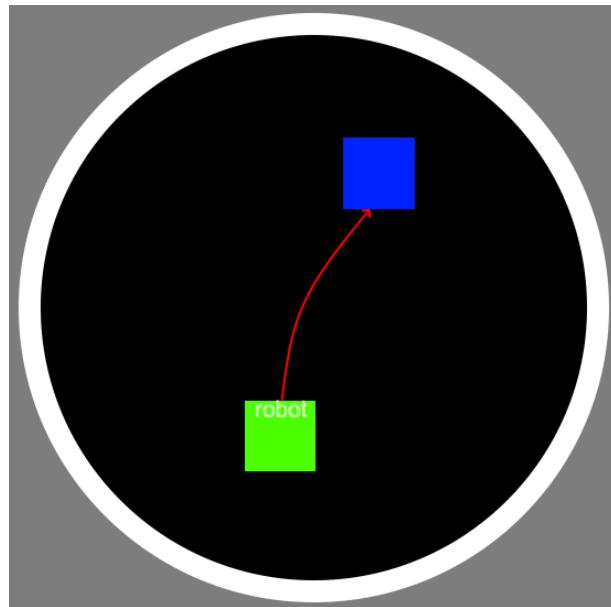
**Figure 23 : déplacement vers le robot en se déviant vers la gauche**

Sur cette figure nous voyons que notre robot va avancer sur le robot adverse en se déplaçant sur la gauche pour avant directement sur celui-ci. Pour faire ce mouvement ont utilisé le moteur de gauche à 100% et le moteur de droite à (100-distance)%.

Nous prenons le paramètre de la distance qui nous sépare du robot pour ajuster notre déplacement;

- **void deriverDroit(int distance)**

cet fonction reçoit un paramètre qui est la distance que le capteur voix et envoyé aucun paramètre



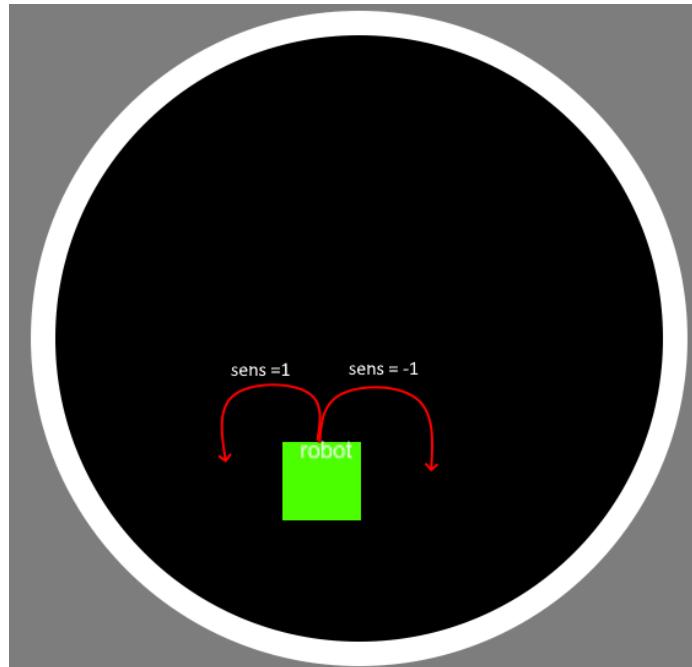
**Figure 24 : déplacement vers le robot en se déviant vers la gauche**

Sur cette figure nous voyons que notre robot va avancer sur le robot adverse en se déplaçant sur la droite pour avant directement sur celui-ci. Pour faire ce mouvement ont utilisé le moteur de gauche à 100% et le moteur de droite à (100-distance)%

Nous prenons le paramètre de la distance qui nous sépare du robot pour ajuster notre déplacement.

- **void tourner(bool sens)**

cet fonction reçoit un paramètre qui le sens et envoyé aucun paramètre



**Figure 25 : Rotation du robot**

Sur cette figure, nous voyons que le robot tourne sur lui-même selon le sens que nous lui donnons (-1 vers la droite, 1 vers la gauche).

## 2.3 Conclusion de la conception préliminaire du produit

En conclusion, nous avons réussi à répondre à toutes les exigences demandées par le client présentes sur le cahier des charges.

Nous pouvons donc passer à la phase de conception détaillée.

### 3. Conception détaillée du produit

Ce chapitre détaille la conception du produit développé. Il constitue une preuve de la conformité du produit. Chaque paragraphe de cette étude fait donc clairement référence aux exigences client issues du [CDC].

#### 3.1 Conception détaillée du robot mini-sumo

#### 3.2 Conception détaillée de la partie acquisition

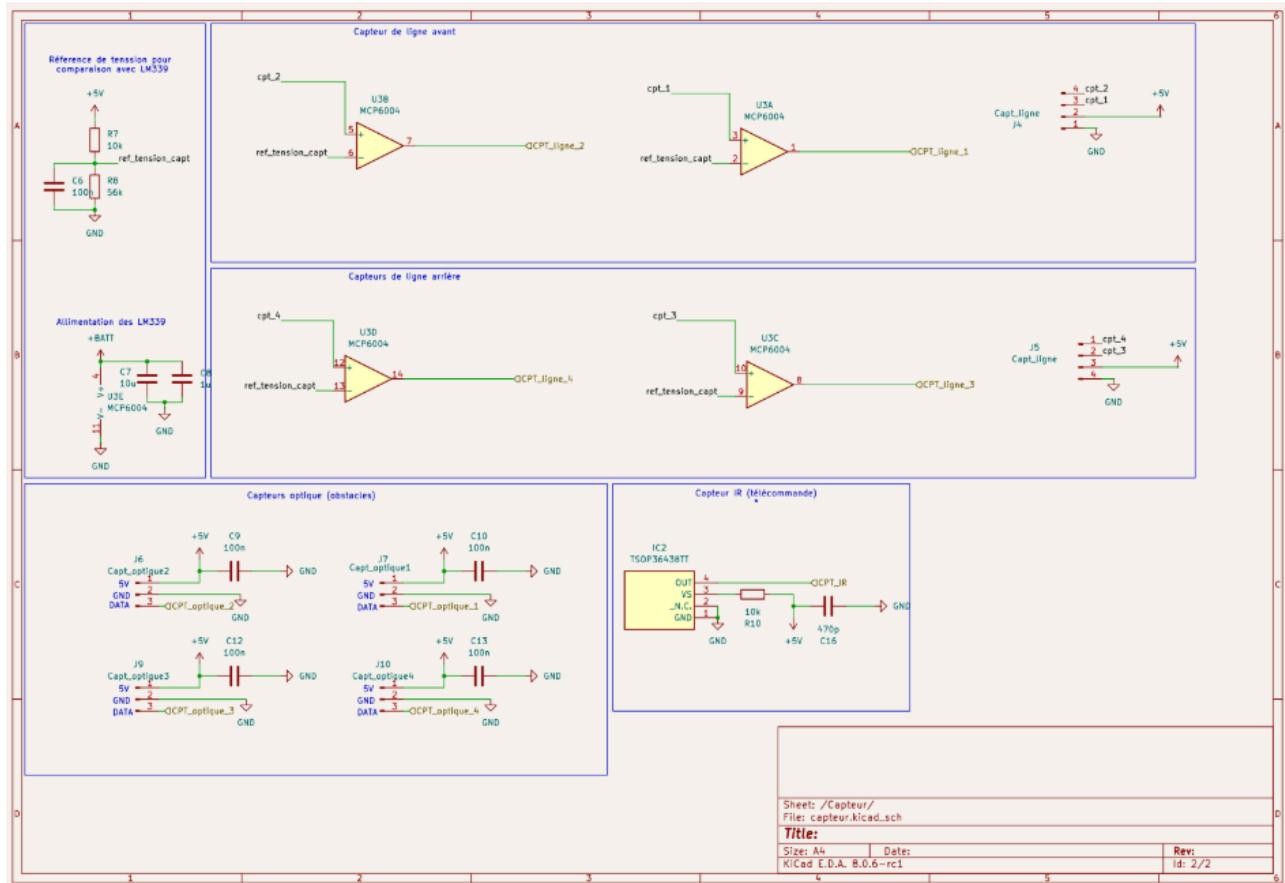


Figure 26 : Schéma électrique détaillé partie acquisition

### 3.2.1 Capteurs adversaires

**Référence de conception :** CDT\_EXIG\_ADVERSAIRES

**Exigences client vérifiées :** EXIG\_ADVERSAIRES

#### 3.2.1.1 Schéma électrique des capteurs adversaires

Pour notre stratégie des capteurs adversaires, nous allons utiliser 4 capteurs optiques, dont 2 composants différents :

- 3 capteurs optiques Sharp GP2Y0A41SK0F mesurant une distance entre 4 et 30 cm d'après leur datasheet à la page 6. On catégorise ces capteurs de "court"
- 1 capteur optique Sharp GP2Y0A021YK mesurant une distance entre 10 et 80 cm d'après leur datasheet à la page 3. On catégorise ce capteur de "long"

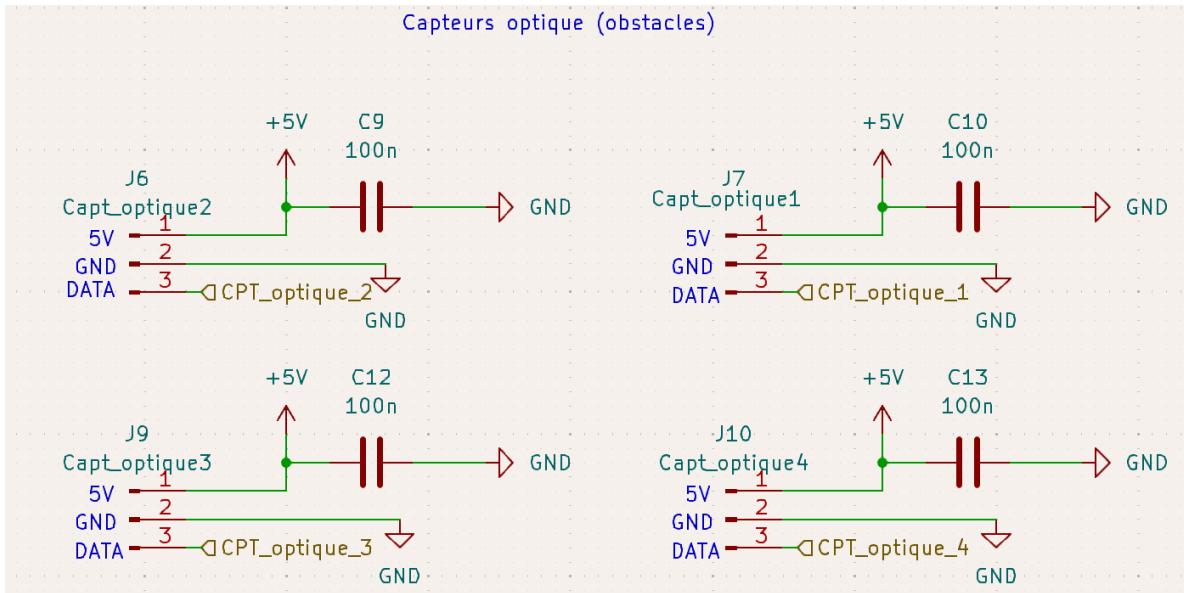
Nous avons choisi cette stratégie pour avoir la meilleure visibilité à courte distance avec 3 capteurs optiques "court". Le but du capteur optique "long" est de pouvoir détecter le robot s'il n'est plus assez proche de nous, donc non perceptible par les capteurs "court".

D'après la datasheet des capteurs optiques, leur tension d'alimentation est comprise entre 4.5V et 5.5V. Nous avons donc choisi de les alimenter par une tension de 5V car c'est la tension commune acceptée par la majorité des composants de notre système.

Pour le branchement de ces capteurs, nous avons besoin à chacun de leur fournir une connexion à la masse ,une alimentation, ici de 5V, et une connexion pour récupérer les données analogiques envoyées. Nous aurons donc besoin de connecteurs à 3 broches pour chaque capteur.

De plus, nous ajouterons un condensateur en parallèle sur la broche d'alimentation de chaque capteur. Ces condensateurs ont comme but d'assurer le bon fonctionnement des capteurs optiques. Leur but est de fournir de l'énergie si le capteur à une grande demande d'énergie en très peu de temps.

Nous avons donc opté pour le schéma électrique ci-contre pour les capteurs optiques :



**Figure 27 : Schéma électrique capteurs adversaires**

Pour la valeur des condensateurs, nous avons ajouté des condensateurs de 100nF entre la tension 5 V et la masse pour stabiliser l'alimentation locale du circuit. Nous avons sélectionné cette valeur après discussions avec notre superviseur de projet.

Ces condensateurs servent à filtrer les bruits à haute fréquence garantissant une alimentation au capteur stable et propre.

### 3.2.1.2 Code informatique des capteurs adversaires

#### Fonction uint8\_t lireCapteurOptiqueCourt(uint8\_t broche)

Entrée : Prend en entrée la broche du composant du capteur optique. On utilise une variable `uint8_t` car la broche est positive, entière et la valeur des broches est inférieure à 255.

Sortie : Renvoie la valeur de la distance en cm entière. On utilise une variable `uint8_t` pour les mêmes raisons que l'entrée

Fonction réalisée : Convertit la tension envoyée par le capteur optique en une distance en cm

#### Fonction uint8\_t lireCapteurOptiqueLong()

Entrée : Prend aucune entrée car la fonction est seulement destinée au capteur optique long du produit.

**Sortie :** Renvoie la valeur de la distance en cm entière. On utilise une variable `uint8_t` pour les mêmes raisons que la fonction précédente

**Fonction réalisée :** Convertit la tension envoyée par le capteur optique long en une distance en cm.

### 3.2.2 Capteurs de sol

**Référence de conception :** CDT\_CAPTEUR\_SOL

**Exigences client vérifiées :** EXIG\_LUMINOSITE

#### 3.2.2.1 Schéma électrique des capteurs de sol

Lors de nos tests sur les capteurs de contrastes, le capteur a renvoyé une tension de 4.78 V pour le noir et 3.85 V pour le blanc. Nous avons ainsi établi une valeur de référence qui nous permet d'identifier avec certitude que la couleur du dohyo est noire lorsque la tension mesurée dépasse ce seuil.

Pour la détection des contrastes, nous avons choisi d'utiliser un comparateur MCP6004. Nous avons sélectionné ce composant car c'était le seul composant embarquant 4 comparateurs en lui-même et en CMS.

Ce composant nous permettra de comparer la tension obtenue en fonction de la couleur du dohyo à une tension de référence suffisamment élevée. Ainsi, lorsque la couleur commence à s'approcher du blanc, le comparateur changera d'état, signalant un passage du noir au blanc.

Nous avons choisi de prendre un comparateur car les capteurs de contrastes renvoient des valeurs analogiques, or nous n'avions plus assez de port analogiques sur notre carte Arduino dû aux autres capteurs du système. De plus, le résultat que nous voulons recevoir est binaire, soit 0 ou 1. Donc, pour l'économie de broches analogiques, nous avons décidé d'utiliser des comparateurs avec les capteurs de contrastes.

Pour créer notre tension de référence nous faisons un pont diviseur de tension, sachant que nous voulons 4.26V pour `ref_tension_capt` en sachant que  $V_{cc} = 5V$ , nous devons dimensionner les 2 résistances constituant ce point diviseur.

Nous devons prendre  $V_{cc} = 5V$  et non la tension apportée par la batterie car celle-ci peut fluctuer en fonction de la charge, ce qui n'est pas le cas par celle transmise par le régulateur linéaire. Si notre tension d'alimentation du pont diviseur n'est pas stable, notre tension de sortie ne sera pas constante et donc les données renvoyées par les comparateurs pourraient être faussées.

Dans un premier lieu nous avions poser  $R7=1k\Omega$ :

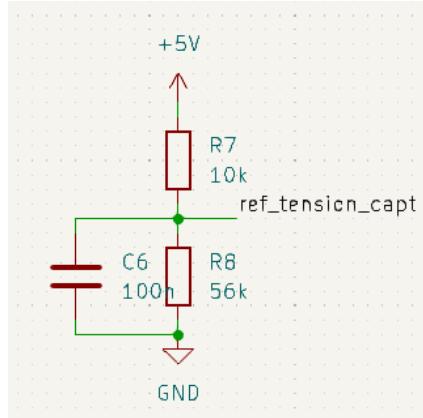
$$R8 = \frac{R7 \cdot V_{cc}}{V_{ref}} - R7 = \frac{1 \cdot 10^3 \cdot 5}{4.26} - 1 \cdot 10^3 = 173.7 \Omega \text{ donc } R8_N = 180\Omega \text{ série E96}$$

Or, avec ces résistances, nous consommons un courant assez important. Nous pouvons éviter

cette surconsommation en augmentant les valeurs des 2 résistances tout en gardant la même tension de référence.

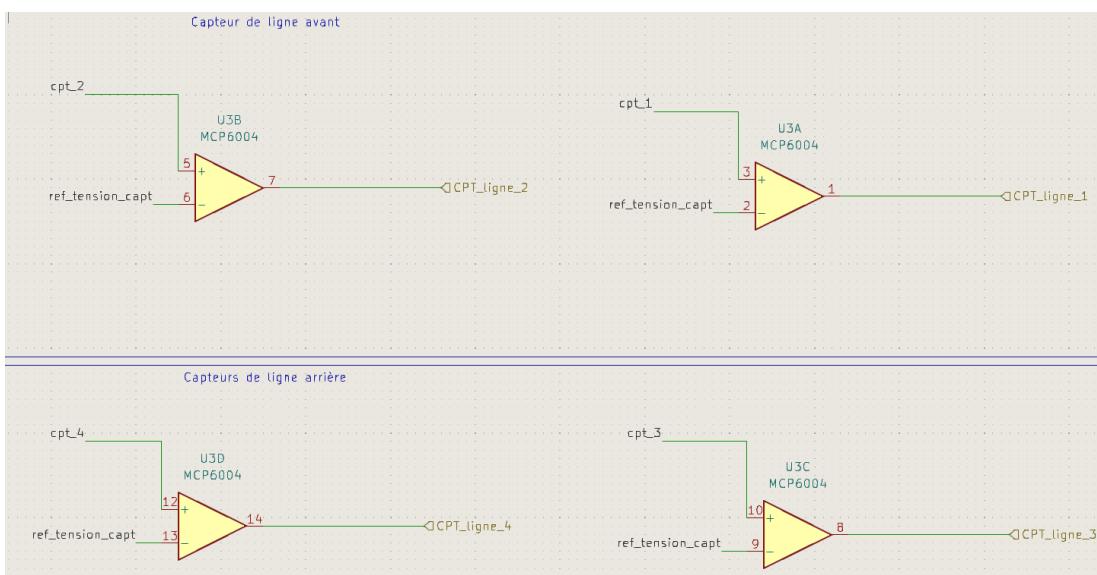
Donc nous avons regardé les résistances misent à notre disposition et avons décidé de les augmenter d'un rapport x56.

Ce qui donne :  $R8=56k\Omega$  et  $R7=10080\Omega$  et leur valeur normalisé donne :  $R8_N = 56k\Omega$   $R7_N = 10k\Omega$ .



**Figure 28 : Schéma électrique référence tension pour comparaison de tension**

Puis nous avons ajouté un condensateur de découplage de 100 nF autour de la résistance branché à la masse pour stabiliser le signal et réduire les éventuelles perturbations dues aux bruits électriques améliorant ainsi la fiabilité de la référence.

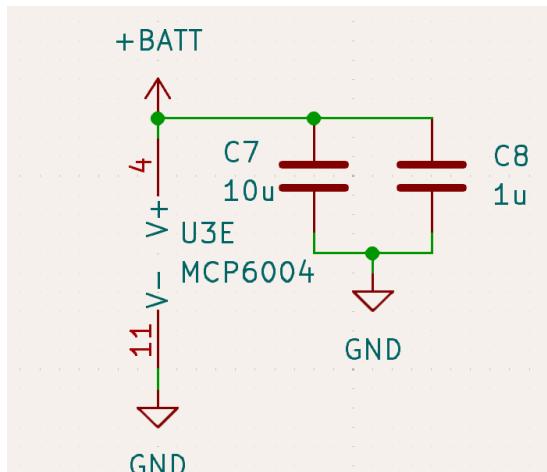


**Figure 29 : Schéma électrique comparateur MCP6004**

Pour alimenter notre comparateur MCP6004, nous avons choisi une tension de 7.4 V (+BATT). Donc, selon la datasheet du comparateur, cette alimentation garantit une tension de sortie maximale d'environ  $7.4 - 2.5 = 4.9$  V.

Si nous avions utilisé une alimentation de 5V, la tension de sortie aurait été limitée à environ 2.5V, ce qui se situe dans la zone intermédiaire des seuils logiques interprétés par la fonction "digitalRead()" de l'Arduino. En effet, pour des entrées numériques fonctionnant en 5V une tension inférieure à 1.5 V est interprétée comme un niveau bas tandis qu'une tension supérieure à 3 V est reconnue comme un niveau haut. Une sortie proche de 2.5 V aurait risqué d'être mal interprétée ou de causer des instabilités dans la lecture.

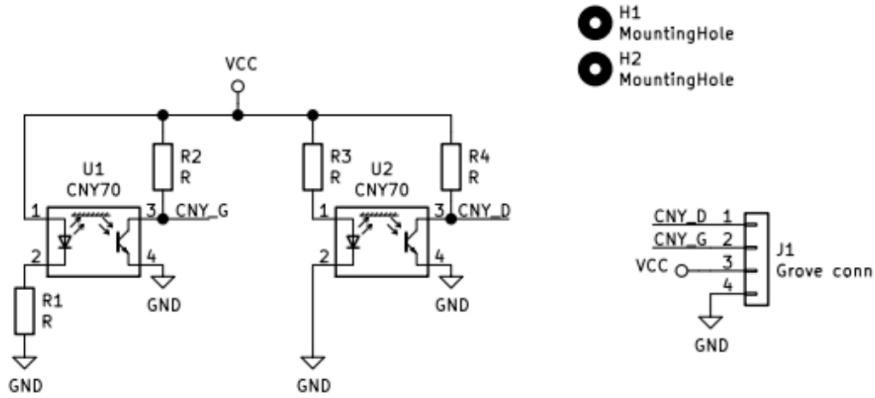
En alimentant le comparateur en 7.4 V, nous obtenons une sortie d'environ 4.9 V bien au-delà du seuil de 3 V requis pour un niveau haut ainsi la détection sera fiable et sans ambiguïté pour l'Arduino.



**Figure 30 : Schéma électrique alimentation des MCP6004**

De plus, nous avons ajouté des condensateurs de découplage autour du comparateur de  $10\ \mu F$  et  $1\ \mu F$ , le premier pour stabiliser l'alimentation en atténuant les variations lentes ou fluctuations de tension et le second pour filtrer les hautes fréquences et supprimer les bruits parasites améliorant ainsi la précision du comparateur.

Pour les capteurs de contraste nous utiliserons le PCB déjà réalisé suivant :



**Figure 31 : Schéma électrique capteurs de contraste**

Avec CNY\_D et CNY\_G dans le schéma ci-dessus les valeurs de sortie des capteur sont équivalentes à cpt1 et cpt2 dans notre schéma électrique.

Nous avons réalisé le dimensionnement des résistances à l'aide de la datasheet du capteur de contraste :

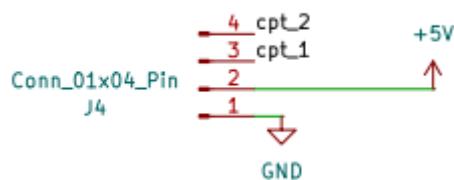
Nous choisissons le courant traversant la diode  $I_F = 10\text{mA}$  donc graphiquement, on trouve que la tension aux bornes de la diode  $V_F = 1.1\text{V}$  à l'aide de la figure 3 de la datasheet.

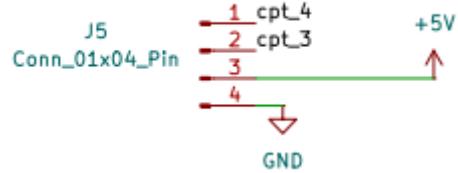
$$\text{Donc } R = \frac{U}{I} = \frac{V_{cc} - V_f}{I_f} \Leftrightarrow R_1 = R_3 = \frac{5 - 1.1}{10 \cdot 10^{-3}} = 195 \Omega$$

Pour le calcul de  $R_2 = R_4$  dans le schéma ci-dessus, on trouve à l'aide de la datasheet page 2 dans les *basic characteristics* que le courant traversant le collecteur du transistor doit être  $I_c = 1\text{mA}$

$$\text{Donc } R = U/I \Leftrightarrow R_2 = R_4 = \frac{5}{1 \cdot 10^{-3}} = 5000 \Omega$$

Les résistances sont directement intégrées au PCB donc notre schéma électrique pour les capteurs contrastes est le suivant :





**Figure 32 : Schéma électrique pour capteurs contrastes**

Donc, pour connecter le PCB des capteurs de contrastes, nous avons utilisé 2 connecteurs à 4 broches pour connecter les 4 capteurs de contrastes de notre système

### 3.2.2.2 Code informatique des capteurs de sol

**Fonction :** `bool lireCapteurContraste(uint8_t broche)`

**Entrée :** Cette fonction prend en entrée la broche d'un des capteurs de contraste. On utilise une variable `uint8_t` car la broche est positive, entière et la valeur des broches est inférieure à 255.

**Sortie :** Retourne une valeur booléenne 1 ou 0 respectivement pour un contraste noir ou blanc.

**Fonction réalisée :** Lit l'état logique du capteur pour déterminer le contraste détecté.

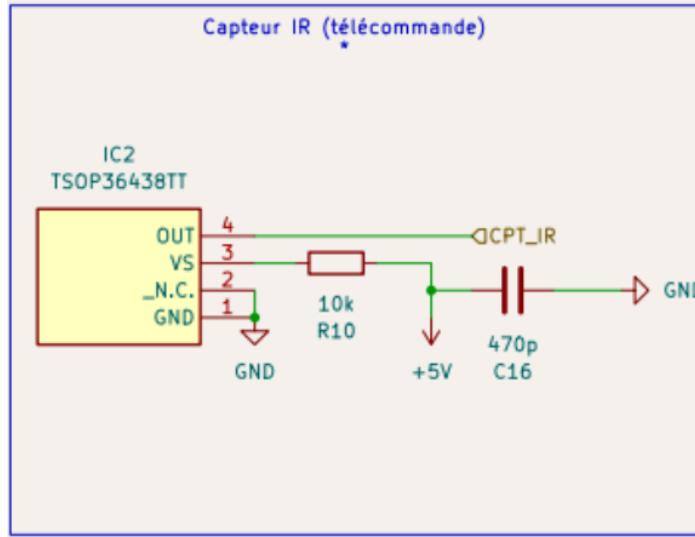
### 3.2.3 Capteurs de télécommande

**Référence de conception :** CDT\_CAPTEUR\_IR

**Exigences client vérifiées :** EXIG\_DEPART

#### 3.2.3.1 Schéma électrique des capteurs de télécommande

Nous avons pris le capteur IR TSOP36438TT pour remplir l'exigence de départ du robot car c'était le seul disponible dans nos stocks lors de la conception et qu'il remplit les conditions.

**Figure 33 : Schéma électrique capteur IR**

Pour utiliser le capteur IR TSOP36438TT, nous avons dimensionné une résistance et un condensateur selon les recommandations de sa datasheet.

Nous avions un PCB avec le capteur IR déjà réalisé auparavant, nous avons donc décidé de garder la valeur du condensateur et de la résistance utilisée. Nous prenons alors une résistance de 10 kΩ pour stabiliser l'état logique de la sortie du capteur et un condensateur de découplage de 470pF entre Vcc et la masse pour filtrer les bruits d'alimentation.

Nous connectons la donnée renvoyée par le capteur infrarouge à une broche digitale de l'Arduino. Malgré le fait que le capteur infrarouge renvoie une valeur analogique, lorsqu'il reçoit une quelconque donnée infrarouge, il renvoie une tension supérieure à 2.5V. Nous pouvons donc simplement utiliser la fonction `digitalRead()` d'Arduino pour détecter si nous recevons une quelconque donnée infrarouge car cette fonction renvoie 1 si la tension à la broche est supérieur à 2.5V.

### 3.2.3.2 Code informatique des capteurs de télécommande

#### Fonction bool lireIR()

Entrées : Ne prend aucune variable en entrée

Sorties : Renverra une valeur booléenne, c'est-à-dire soit true, correspondant à 1, ou false, correspondant à 0.

Fonction réalisée : Ne fais rien tant que nous ne recevons aucun signal infrarouge. Après la réception d'un quelconque, la fonction renvoie `true`

### 3.3 Conception détaillée de la partie action

#### 3.3.1 Pont en H

Référence de conception : CDT\_PONT\_EN\_H

Exigences client vérifiées : EXIG\_DEPLACEMENT

##### 3.3.1.1 Schéma électrique du pont en H

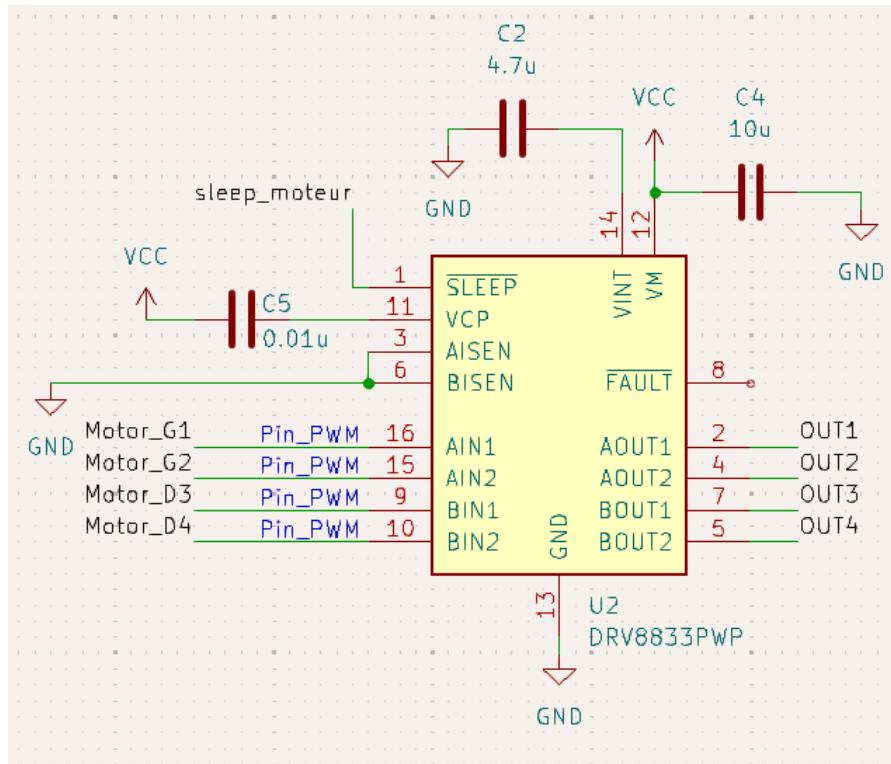


Figure 34 : schéma électrique du pont en H

Comme dit précédemment dans le dossier de conception préliminaire, nous utilisons seulement la puce DRV8833 et non le Pololu. Nous avons donc dû ajouter sur notre schéma électrique, grâce à la datasheet du DRV8833, des condensateurs de découplage.

La broche "nSLEEP" est connectée en sortie du comparateur de la partie énergie permettant la protection de nos moteurs en éteignant ceux-ci si la tension d'alimentation de la batterie est trop faible. Nous avons fait ce choix car les moteurs sont les composants qui consomment le plus dans notre robot.

Nous avons connecté les broches "AISEN" et "BISEN" à la terre car nous n'utilisons pas ces broches permettant de mesurer l'intensité utilisée pour le contrôle et l'alimentation des moteurs. Et finalement, la broche "nFAULT" n'est connectée à rien car celle-ci nous est inutile.

### 3.3.1.2 Code informatique du pont en H

Nous avons ainsi rédigé une fonction par moteur ainsi qu'une fonction contrôlant l'éclairage de la LED bleue. Ainsi, nos fonctions s'appellent VITESSE\_MOT\_G pour le contrôle du moteur gauche, VITESSE\_MOT\_D pour le contrôle du moteur droit et ETAT\_LED pour gérer l'éclairage de la LED bleue permettant de signaler à l'arbitre que le signal de la télécommande a bien été reçu.

Ainsi nous avons besoin des sorties suivantes sur les pins de la carte arduino ( les numéros de pin ne sont pas définitifs et n'ont servi qu'au dérisquage du code informatique ).

Ensuite, nous avons utilisé les datasheets du DRV8833 ainsi que de la Pololu utilisant ce même composant pour contrôler correctement chacun des deux moteurs avec les différentes fonctions citées précédemment.

Pour la formule permettant de gérer la vitesse, nous avons multiplié la vitesse en pourcentage reçue par +/- 2.55 car les valeurs du PWM de l'arduino vont de 0 à 255. Or nous avons en entrée une valeur en pourcentage de 0 à 100.

Lorsque le pourcentage est positif, la roue tourne vers l'avant du robot et inversement lorsque la valeur est négative. Ainsi la valeur "vitesse" peut prendre n'importe quelle valeur entre -100 et 100. De plus, nous avons également enlevé toutes les valeurs entre -20 et 20 car après avoir effectué des essais, nous nous sommes rendu compte que pour des si petites valeurs, le moteur n'était pas suffisamment alimenté pour déplacer le robot.

### 3.3.2 Diodes électroluminescentes

**Référence de conception :** CDT\_LED

**Exigences client vérifiées :** sans objet

#### 3.3.2.1 Schéma électrique des LED

Pour la LED verte qui va servir à indiquer si le robot est sous tension ou non, nous avons choisi de prendre la LED verte ASMT-UGB5-NV702.

Nous devons maintenant dimensionner la résistance associée pour que la LED verte brille suffisamment pour que celle-ci soit visible mais qu'elle ne dérange pas au regard.

Pour cela nous poursuivons par tâtonnement en augmentant peu à peu la résistance afin de diminuer l'intensité lumineuse. Suite au dérisquage, nous avons obtenu une valeur de  $3900\Omega$ .

Selon la datasheet de la LED, l'intensité du courant traversant la LED est inférieure à 10mA.

En normalisant cette résistance grâce au HTUT 1 vu en première année, on trouve une valeur de résistance normalisée de  $R = 3900 \Omega$  Série E24 (+/-5%) en boîtier 1206.

Pour la LED bleue qui va servir à indiquer si le robot reçoit la trame envoyée par la télécommande, nous avons choisi de prendre la LED bleue ASMT-UBB5-NS8Q2.

Nous devons maintenant dimensionner la résistance associée pour que la LED bleue brille suffisamment pour que celle-ci soit visible mais qu'elle ne dérange pas au regard.

Pour cela nous poursuivons par tâtonnement en augmentant peu à peu la résistance afin de diminuer l'intensité lumineuse. Suite au dérisquage, nous avons obtenu une valeur de  $470\Omega$ .

Selon la datasheet de la LED, l'intensité du courant traversant la LED est inférieure à 10mA.

IUT Bordeaux Département GEII	Référence : RMS_DDC_EQ11 Révision : 3 – 04/11/2024	38/56
----------------------------------	---	-------

En normalisant cette résistance grâce au HTUT 1 vu en première année, on trouve une valeur de résistance normalisée de  $R = 470 \Omega$  Série E96 ( $\pm 5\%$ ) en boîtier 1206.

### 3.3.2.2 Code informatique de la LED bleue

Nous avons ainsi rédigé une fonction qui contrôle l'éclairage de la LED bleue se nommant ETAT\_LED pour gérer l'éclairage de la LED bleue permettant de signaler à l'arbitre que le signal de la télécommande a bien été reçu.

Tout d'abord nous devons déclarer la pin associée à la LED bleue. Puis nous réalisons la fonction. La fonction prend en entrée un booléen (1 ou 0) permettant d'allumer la LED (1) ou de l'éteindre (0).

## 3.4 Conception détaillée de la partie énergie

### 3.4.1 Sécurisation de la batterie

**Référence de conception :** CDT\_SECUR\_BATT

**Exigences client vérifiées :** EXIG\_SECUR\_BATT

Afin de répondre au mieux à l'exigence de sécurisation de la batterie, nous avons fait le choix de passer par un système électronique, cela nous permet de nous assurer que peu-importe l'état dans lequel se trouve le MCU nous pouvons couper l'alimentation des moteurs.

Afin de réaliser l'étage de sécurisation, nous avons utilisé un comparateur.

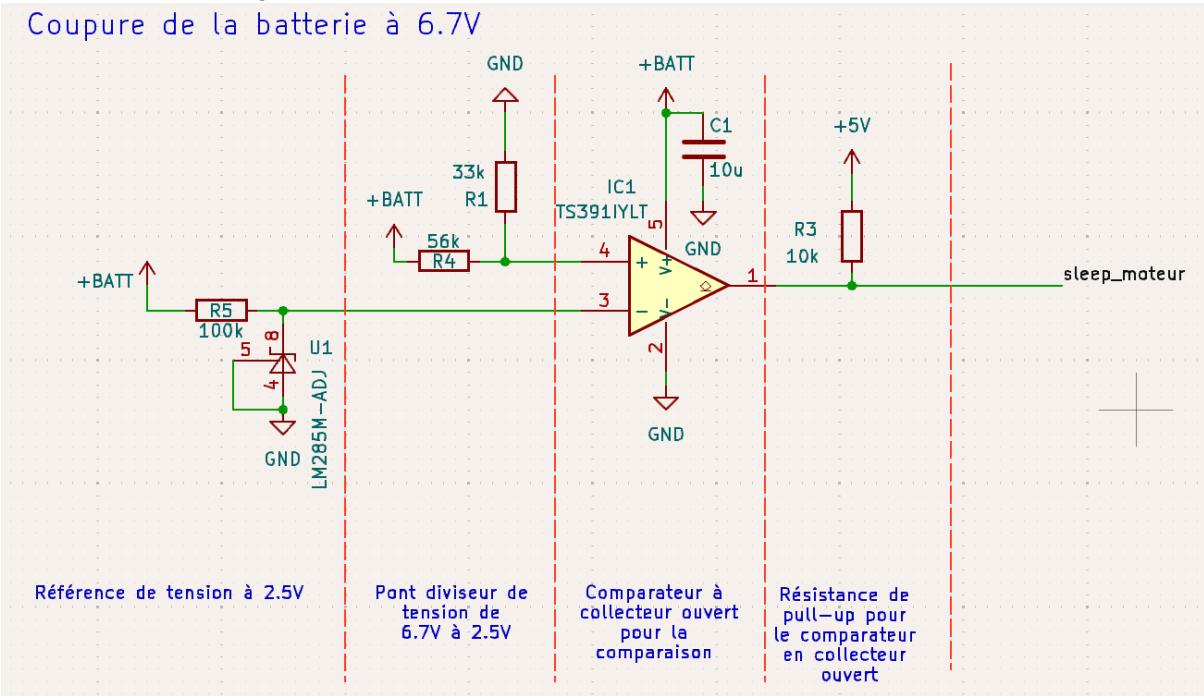


Figure 35 : Schéma électrique gestion de la coupure de la batterie à 6.7V

**Référence de tension à 2.5V :**

Afin de créer une tension de référence stable nous allons utiliser un composant qui s'appelle un référencement de tension, la référence de tension fonctionne comme une diode, lorsqu'un courant la traverse elle permet de créer une différence de potentiel à ces bornes. La principale différence entre une diode et la référence de tension est la précision de la tension à ces bornes.

Au sein du département GEII nous avons à notre disposition des références de tension de 1.2 V et 2.5 V, nous avons choisi la référence de tension de 2.5 V afin d'avoir plus de précision dans la comparaison à venir.

Nom de la référence de tension : LM385-2.5-N

Afin de fonctionner le LM385-2.5-N doit être alimenté entre 20 µA et 20 mA.

Nous allons faire en sorte que le courant qui traverse la diode soit minimal afin de consommer un minimum.

La valeur max de résistance que nous allons choisir est 100kΩ, ce qui nous permet de prendre la carte dans les mains sans perturber le système électronique.

Le courant induit par une résistance dans le pire des cas (6.7V) :

$$I = \frac{U}{R} = \frac{6.7 - 2.5}{100 * 10^3} = 42\mu A$$

$$42\mu A > 20\mu A$$

Le courant pour un résistance de 100kΩ est supérieur à la valeur minimum de fonctionnement, la résistance est donc adaptée.

**Pont diviseur de tension de 6.7v à 2.5V :**

Nous avons établi une valeur tension fixe précédemment, cette valeur est de 2.5 V, malheureusement nous souhaitons détecter si la tension passe en dessous de 6.7 V et de 2.5 V.

Afin de réaliser cette comparaison nous allons donc descendre la tension de 6.7 V à 2.5V grâce à un pont diviseur de tension.

Afin de déterminer les valeurs de résistance nous partirons du principe que la tension en entrée du pont diviseur est de 6.7V, notre objectif est d'avoir 2.5V en sortie, nous allons donc créer un pont diviseur avec un rapport de 2.68.

Nous fixons une première valeur de résistance à 33 kΩ, la valeur de la seconde résistance est de  $1.68 * 33 k = 55.44 k\Omega$ .

La somme des deux résistances ne dépasse pas 100kΩ nous sommes donc dans des valeurs acceptables.

Les valeurs normalisées des résistances sont 33 kΩ et 56kΩ.

**Comparateur collecteur ouvert pour la comparaison et résistance de pull UP :**

Nous utilisons un comparateur en collecteur ouvert afin d'assurer la comparaison des deux signaux.

Lorsque la tension de la batterie est supérieure à 6.7v alors la sortie est mise à 5V grâce à une résistance de pull UP de 10k.

Sinon elle est mise à la masse.

### Condensateur de découplage du comparateur :

Nous utilisons un condensateur de découplage de  $10\mu F$  de série E3 pour stabiliser la tension du composant, afin d'éviter les chutes de tension et de compenser le fort courant de consommation. Nous avons également choisi ce condensateur pour lisser la tension du régulateur et réduire les bruits parasites. Cette valeur a été choisie à l'aide d'un expert

#### 3.4.2 régulateur de tension

Référence de conception : CDT\_REGULATEUR\_TENSION

Exigences client vérifiées : sans objet

##### 3.4.2.1 Schéma électrique du régulateur de tension

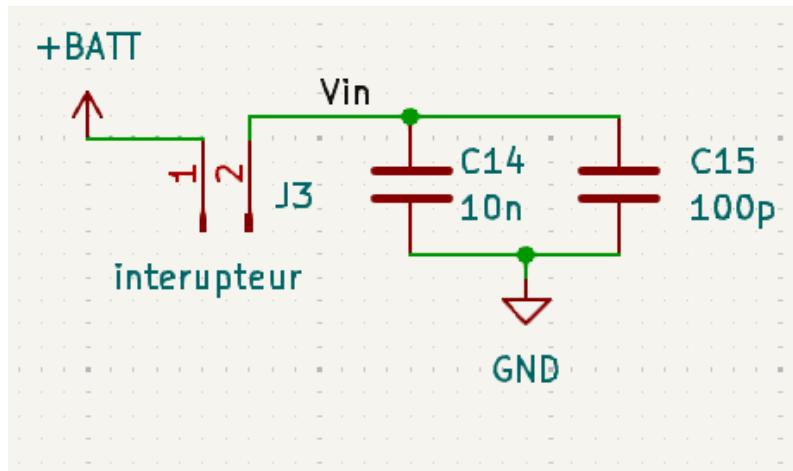


Figure 36 : schéma du régulateur

Pour le régulateur linéaire, nous avons choisi celui déjà inclus dans la carte Arduino UNO, car il correspond à notre besoin. Dans la carte Arduino, il y a un condensateur de  $47\mu F$ , mais ce condensateur ne suffit pas à notre besoin. Nous avons donc ajouté des condensateurs de  $100\mu F$ , de  $10nF$  et de  $100pF$  qui seront de la série E3. Nous avons décidé de mettre ces condensateurs afin de stabiliser la tension selon la fréquence requise par nos composants, ces valeurs sont pris grâce au HTUT 25.

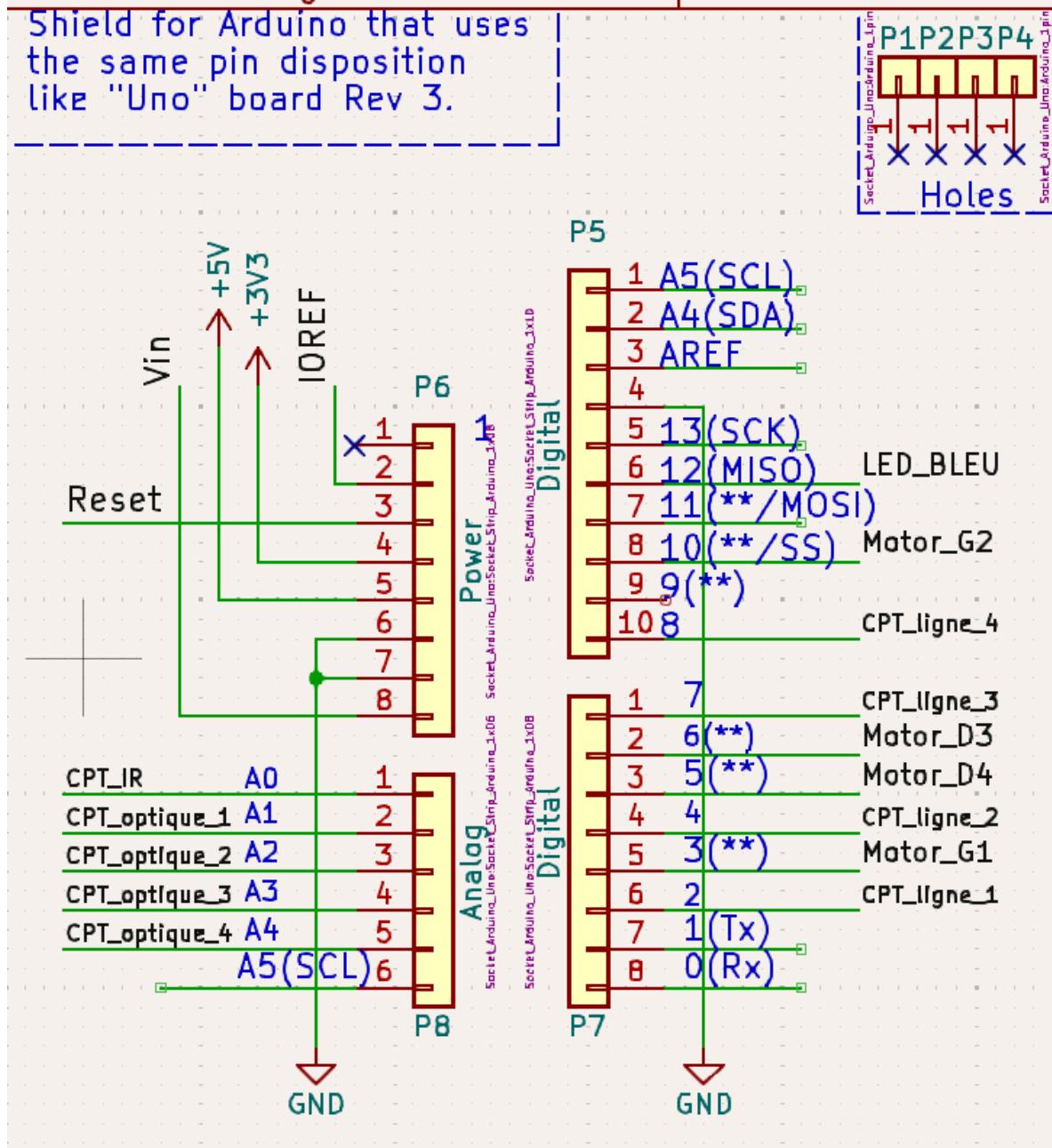
## 3.5 Conception détaillée de la partie traitement

### 3.5.1 Le microcontrôleur ATMEGA328P sur carte de prototypage arduino uno

Référence de conception : CDT\_ATMEGA328P

Exigences client vérifiées : EXIG\_COMPORTEMENT

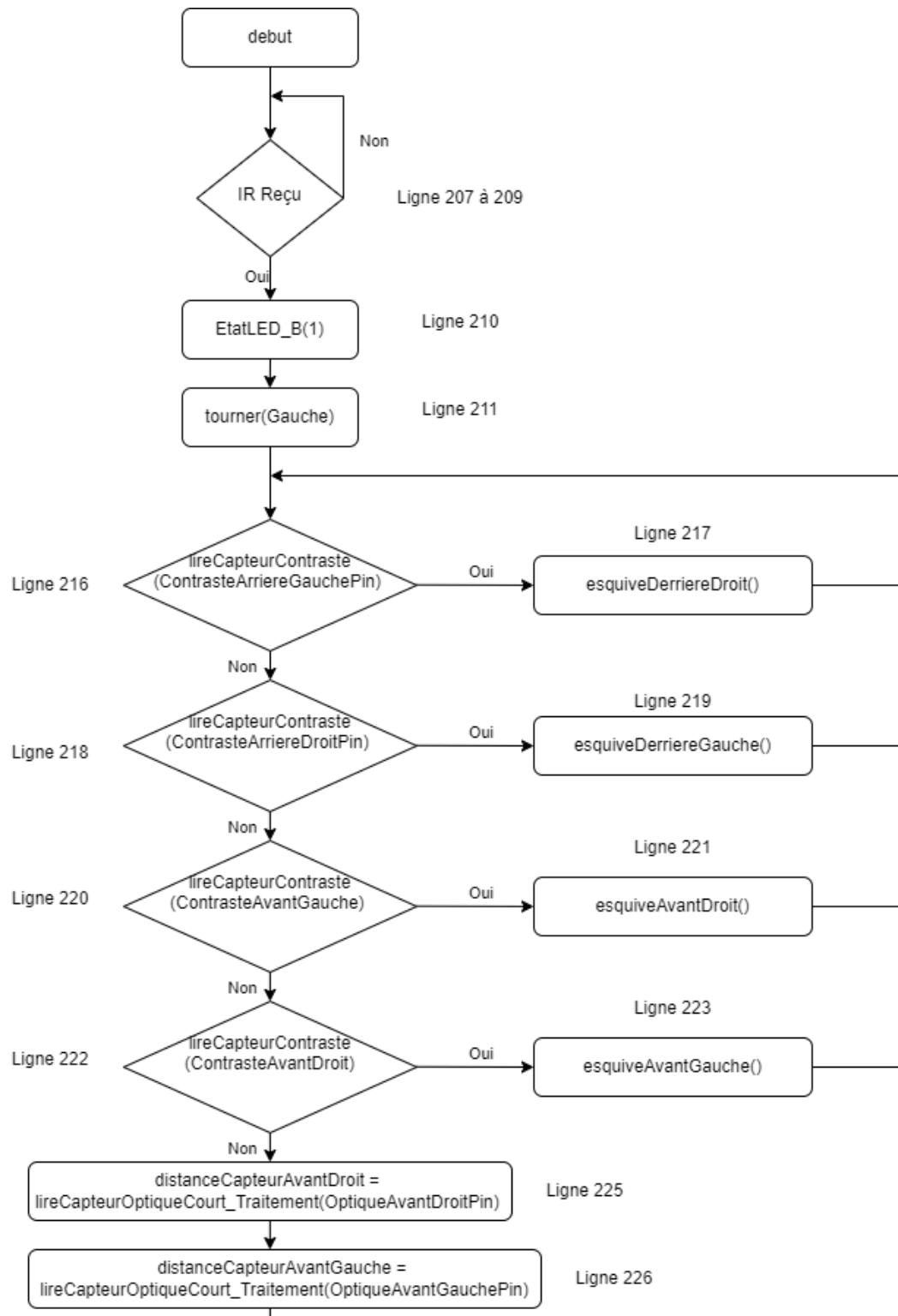
##### 3.4.1.1 Schéma électrique du microcontrôleur

**Figure 37 : schéma du régulateur**

Sur la figure 38 nous voyons que le notre MCU a plusieurs PIN que nous utilisons comme les Pin Analogique qui correspond au capteur optique et le capteur IR, nous avons aussi d'autres

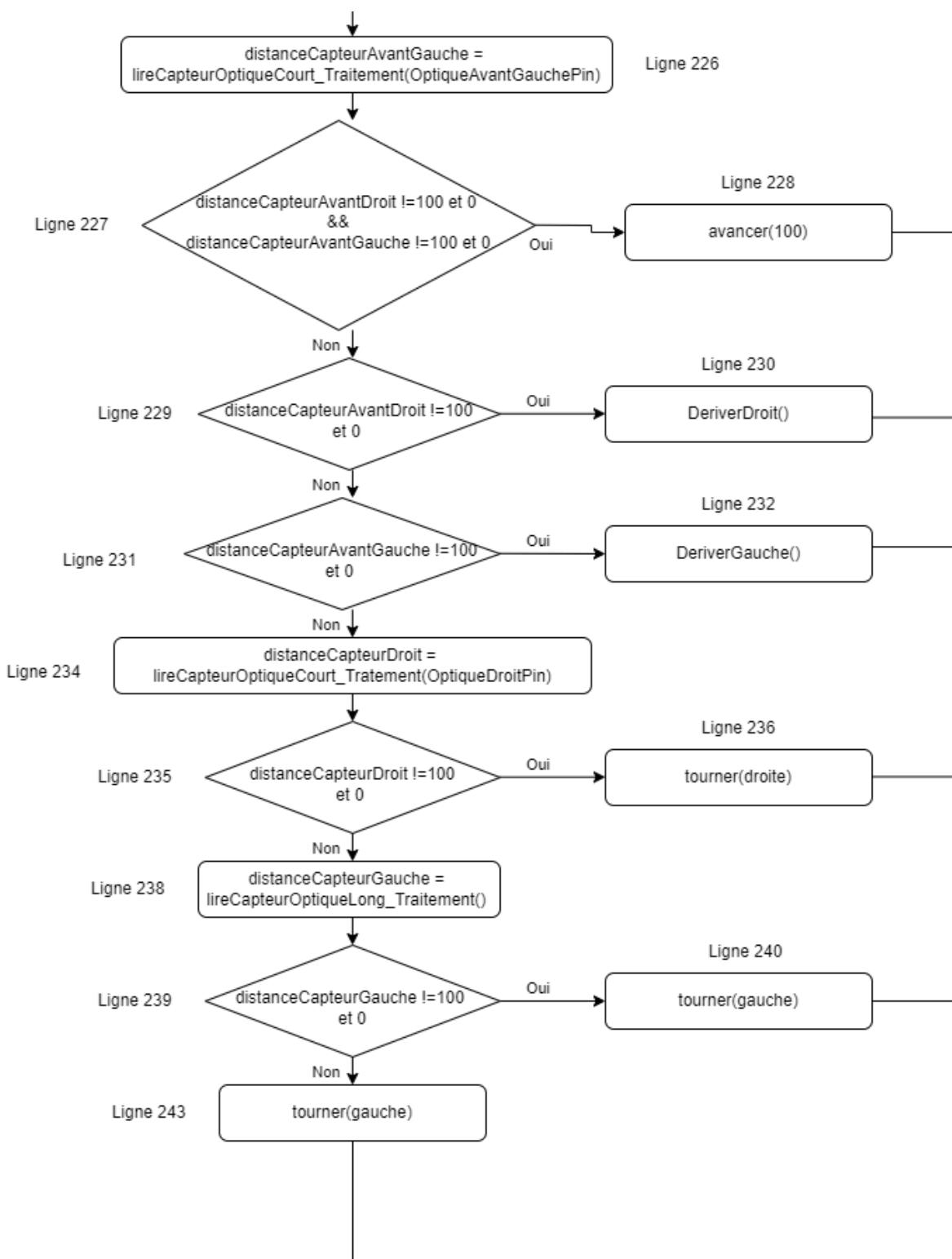
### 3.3.1.2 Code informatique du microcontrôleur

## Robot Mini-Sumo (RMS)



**Figure 38 : synoptique du code informatique**

## Robot Mini-Sumo (RMS)



**Figure 39 : synoptique du code informatique**

Sur cette algorithme nous avons plusieurs bloc ou nous avons mis le ligne correspondant au ligne tu programme

Nous avons rencontré un problème lié aux tests des capteurs optiques. Le problème se manifeste lorsque les capteurs sont trop proches : ils signalent une distance incorrecte en indiquant qu'elle est plus longue. De plus, il est nécessaire de gérer la distance maximale détectable par les capteurs.

Pour résoudre ce problème, nous avons créé deux fonctions :

1. **La fonction lireCapteurCours\_Traitement :**

Cette fonction traite les données des capteurs courts. Elle renvoie **1** si le capteur détecte un objet trop proche et **0** si le capteur détecte une distance trop grande. Cette gestion est spécialement adaptée aux capteurs ayant une portée limitée.

2. **La fonction lireCapteurLong\_Traitement :**

Cette fonction fonctionne sur le même principe que la précédente, mais elle est adaptée aux capteurs longs, capables de mesurer des distances plus importantes. Elle renvoie des valeurs couvrant une plage plus large pour s'adapter à leur portée.

Ensuite, nous avons d'autres fonctions de traitement qui permettent au robot de se déplacer grâce aux moteurs :

1. **La fonction esquiveDerriereGauche :**

Cette fonction utilise les deux moteurs. Le moteur droit est réglé à **-50** et le moteur gauche à **-100**. Cette fonction permet d'esquiver vers la gauche pour éviter une collision lorsque le robot détecte un obstacle avec le capteur arrière droit.

2. **La fonction esquiveDerriereDroit :**

Cette fonction utilise également les deux moteurs. Le moteur droit est réglé à **-100** et le moteur gauche à **-50**. Cette fonction permet d'esquiver vers la droite pour éviter une collision lorsque le robot détecte un obstacle avec le capteur arrière gauche.

3. **La fonction esquiveAvantGauche :**

Cette fonction utilise les deux moteurs. Le moteur droit est réglé à **50** et le moteur gauche à **100**. Elle permet d'esquiver vers la gauche pour éviter une collision lorsque le robot détecte un obstacle avec le capteur avant droit.

4. **La fonction esquiveAvantDroit :**

Cette fonction utilise les deux moteurs. Le moteur droit est réglé à **100** et le moteur gauche à **50**. Elle permet d'esquiver vers la droite pour éviter une collision lorsque le robot détecte un obstacle avec le capteur avant gauche.

5. **La fonction Tourner :**

Cette fonction permet de faire tourner le robot dans un sens ou dans l'autre. Elle prend un paramètre ("droit" ou "gauche") qui détermine le sens de rotation. Ce paramètre ajuste la valeur des moteurs à **100** ou **-100**, inversant ainsi leur sens.

6. **La fonction Avancer :**

Cette fonction permet de faire avancer le robot en réglant les deux moteurs à **100**.

7. **La fonction dériverDroit :**

Cette fonction permet au robot d'avancer tout en se tournant vers la droite. Elle prend en paramètre la distance du robot, qui permet d'ajuster la vitesse des moteurs en fonction de cette distance.

#### 8. La fonction **dériverGauche** :

Cette fonction permet au robot d'avancer tout en se tournant vers la gauche. Elle prend également en paramètre la distance du robot, permettant d'ajuster la vitesse des moteurs selon cette distance.

## 4. Dérisquage des solutions techniques retenues

Ce chapitre détaille les activités de dérisquage des solutions techniques retenues : simulation et/ou prototypage rapide. Il constitue une preuve partielle de la conformité du produit. Chaque paragraphe de l'étude fait donc clairement référence aux exigences client issues du [CDC].

Il permet également de confirmer les résultats théoriques effectués aux paragraphes 2 et 3 en vérifiant le fonctionnement à travers des simulations et/ou prototypages rapides. Pour chaque simulation et/ou prototypage rapide est renseigné le protocole de mise en œuvre. Les résultats des simulations et/ou prototypages rapides sont confrontés aux résultats de l'étude théorique.

L'ensemble des fichiers est disponible dans le dossier : renseignez ici le chemin du dossier où sont situés les fichiers de simulation et/ou prototypage rapide du projet.

### 4.1 Essai programme contrôle des moteurs

**Référence de la simulation :** SIM1

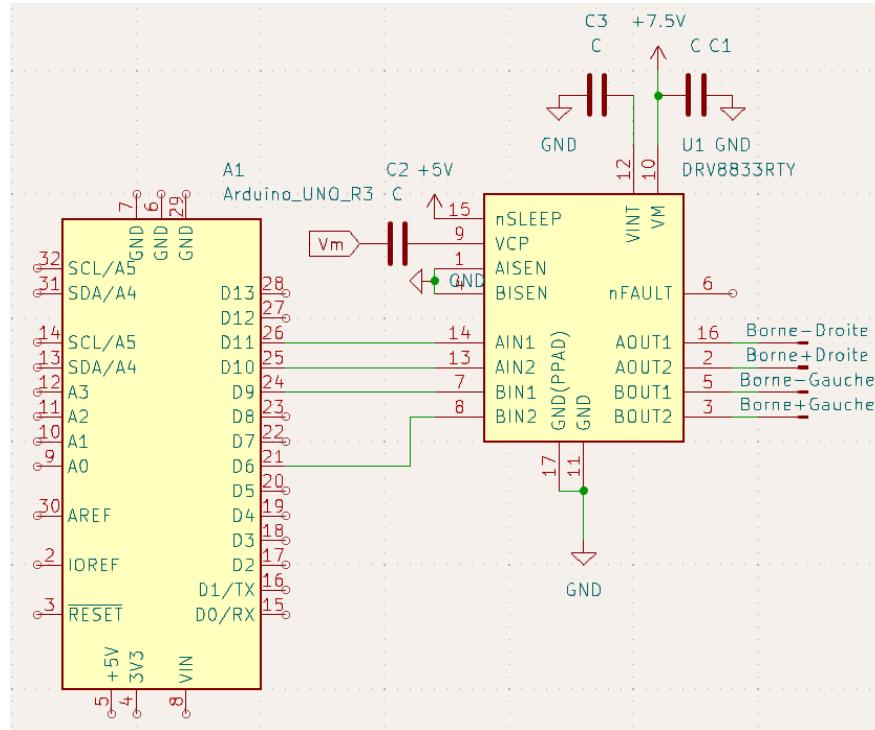
**Exigences client vérifiées :** EXIG\_DEPLACEMENT

**But de l'essai :** Vérifier les différentes comportements du robot en fonction du programme associée

**Fichiers :** [FONCTIONS\\_ACTION.ino](#)

**Procédure de simulation :**

## Robot Mini-Sumo (RMS)



**Figure 40 : schéma électrique de l'essai**

A l'aide d'une breadboard, d'une carte arduino, du robot sumo, une carte pololu DRV8833 utilisant la même puce que celle que nous utiliserons ainsi que d'une alimentation de table réglée à 7,4V nous avons dérisqué les fonctions VITESSE\_MOT\_D et VITESSE\_MOT\_G pour s'assurer de leur bon fonctionnement et ajuster certaines parties du code en cas de nécessité. Pour cela nous avons commencé par écrire un programme test dans le boucle infini du programme faisant appel aux deux fonctions que nous avons créé comme ci-dessous sur la figure 45.

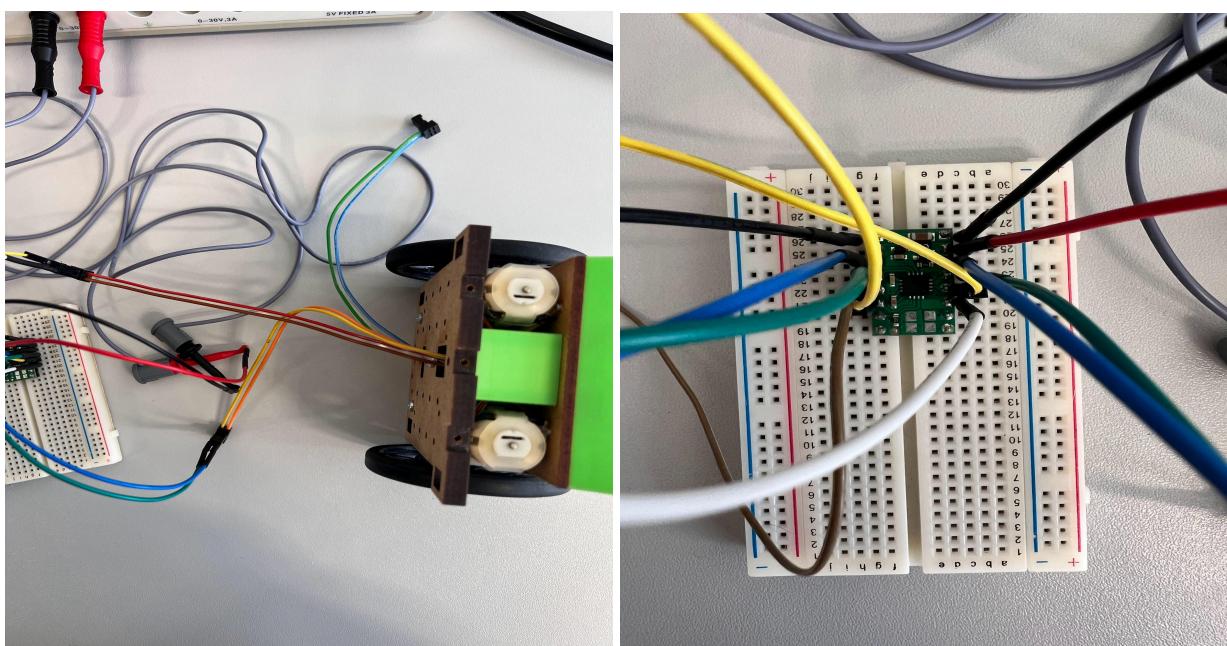
Ensuite, nous avons réalisé le branchement du prototypage en branchant la pololu aux PIN de la carte arduino uno définis sur notre programme. Pour alimenter la pololu, nous utilisons les 5V fournis par l'arduino. L'alimentation de table nous a servi à alimenter les deux moteurs comme nous le montrons sur les figures 45-46-47-48.

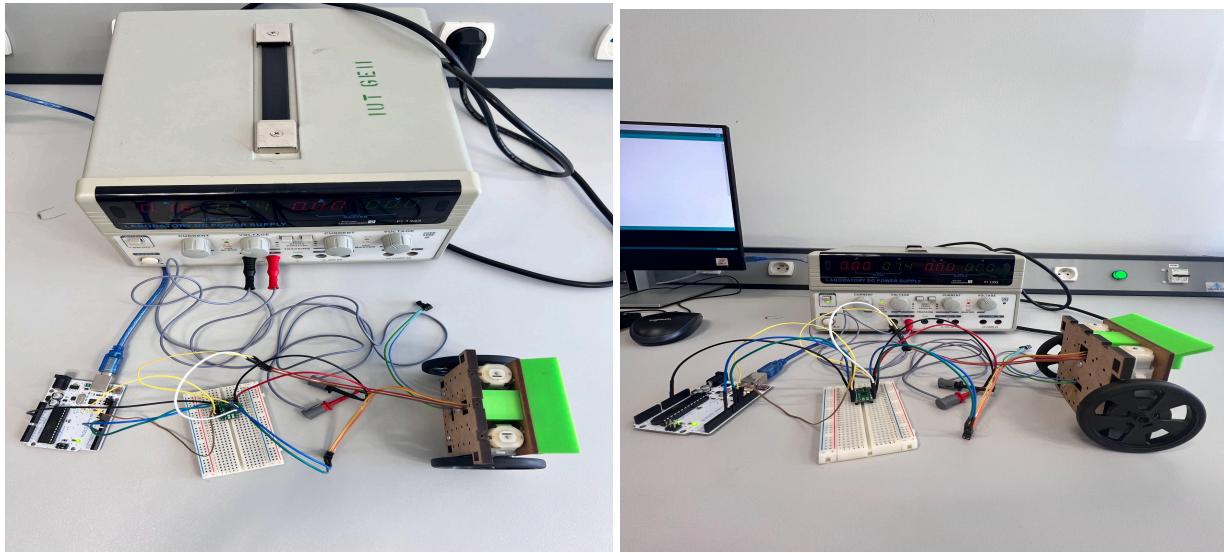
```

61
62 void loop() {
63     //EXEMPLES FONCTIONNEMENT
64     // Avance a 100% pendant 2s
65     VITESSE_MOT_G(100);
66     VITESSE_MOT_D(100);
67     delay(2000);
68     // freinage et arret pendant 2s
69     VITESSE_MOT_G(0);
70     VITESSE_MOT_D(0);
71     delay(2000);
72     // Recule a 50% pendant 3s
73     VITESSE_MOT_G(-50);
74     VITESSE_MOT_D(-50);
75     delay(3000);
76     // Moteur gauche avance a 30% et moteur droit recule a 30% pendant 4s
77     VITESSE_MOT_G(30);
78     VITESSE_MOT_D(-30);
79     delay(4000);
80 }
81

```

Figure 41 : essai des fonctions lors du dérisquage





**Figures 42, 43, 44 et 45 : essai et dérisquage des fonctions d'action**

#### Résultats attendus :

Comme écrit dans le code de test, nous attendions que notre robot se comporte de la manière suivante :

- le robot avance pendant 2 secondes à 100% de sa vitesse maximale
- le robot s'arrête rapidement et reste à l'arrêt pendant 2 secondes
- le robot recule pendant 3 secondes à 50% de sa vitesse maximale
- La roue gauche avance à 30% et la roue droite recule à 30% de leur vitesse maximale (rotation à droite) pendant 4 secondes.

#### Résultats obtenus :

Nous avons obtenu le résultat attendu après plusieurs modifications du code. Les roues tournaient dans le sens voulu et à la vitesse demandée. Nous avons également vérifié le bon fonctionnement des moteurs en mode fast decay pour que le robot soit plus réactif lors des accélérations et des freinages brusques permettant les changements de direction plus efficaces.

#### Statut de l'essai : CONFORME

**Problèmes rencontrés :** Lors de notre tout premier essai, nous avions commis des erreurs au niveau du branchement car la broche positive (+) et la broche négative (-) des moteurs ne sont pas indiqués sur le robot. Nous avons donc corrigé ce problème et indiqué sur les connecteurs et sur le typon quelles sont les broches correspondantes. De plus nous avions programmé le driver en slow decay ce qui rendait le robot moins réactif aux changements brusques tels que les freinages. Nous avons corrigé ce problème en changeant l'état des entrées moteurs analogiques.

## 4.2 Essai programme LED

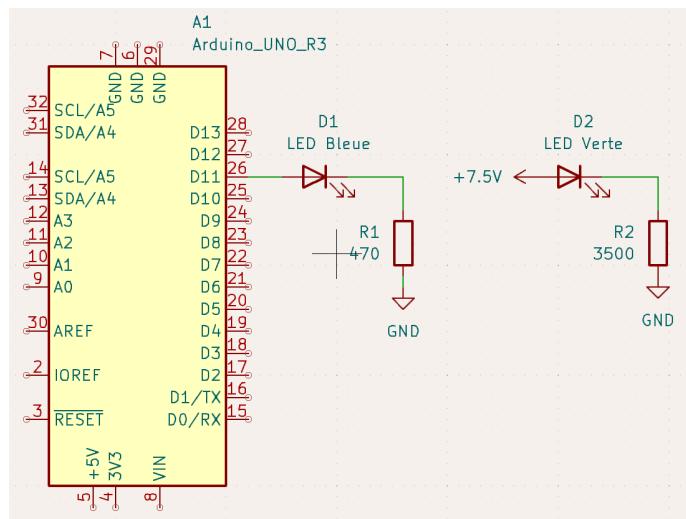
**Référence de la simulation :** SIM2

**Exigences client vérifiées :** sans objet

**But de l'essai :** Vérifier que le programme de la LED bleue fonctionne bien et vérifier que les LED vertes et bleus brillent suffisamment.

**Fichier :** [FONCTIONS\\_ACTION.ino](#)

**Procédure de simulation :**



**Figure 46 : schéma électrique de l'essai des LED**

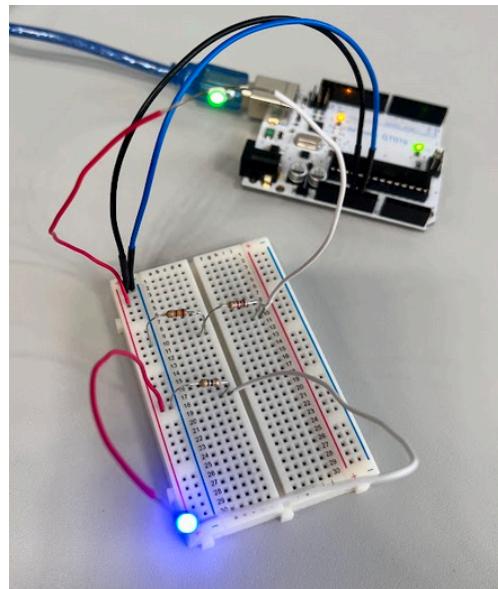
A l'aide d'un breadboard, d'une carte arduino UNO, de la LED bleue et verte, des résistances associées aux LED, et d'une alimentation de table, nous branchons les LED et résistances sur la breadboard suivant les schéma électrique ci-dessus. Puis nous alimentons la LED verte avec l'alimentation de table et nous branchons l'alimentation de la LED bleue sur un pin PWM de la carte arduino.

**Résultats attendus :**

On souhaite que les LED brillent suffisamment mais que cela ne gêne pas la vision des utilisateurs et que la LED bleue s'allume et s'éteint conformément au programme.

**Résultats obtenus :**

IUT Bordeaux Département GEII	Référence : RMS_DDC_EQ11 Révision : 3 – 04/11/2024	50/56
----------------------------------	---	-------



**Figure 47 : photo de l'allumage des LED**

La LED verte brille suffisamment et ne gêne pas lorsqu'on la regarde.

Et après essais avec le programme d'action de la LED bleue, la LED bleue s'allume et s'éteint conformément à notre programme.

**Statut de l'essai : CONFORME**

**Problèmes rencontrés :** Sans objet

### 4.3 Essai programme Capteurs

**Référence de la simulation : SIM3**

**Exigences client vérifiées : EXIG\_ADVERSAIRES, EXIG\_DEPART**

**But de l'essai :** Vérifier le bon fonctionnement des différents capteurs avec un branchement proche du schéma électrique final de la partie acquisition

**Fichier : [Test\\_Cap\\_All.ino](#)**

**Procédure de simulation :**

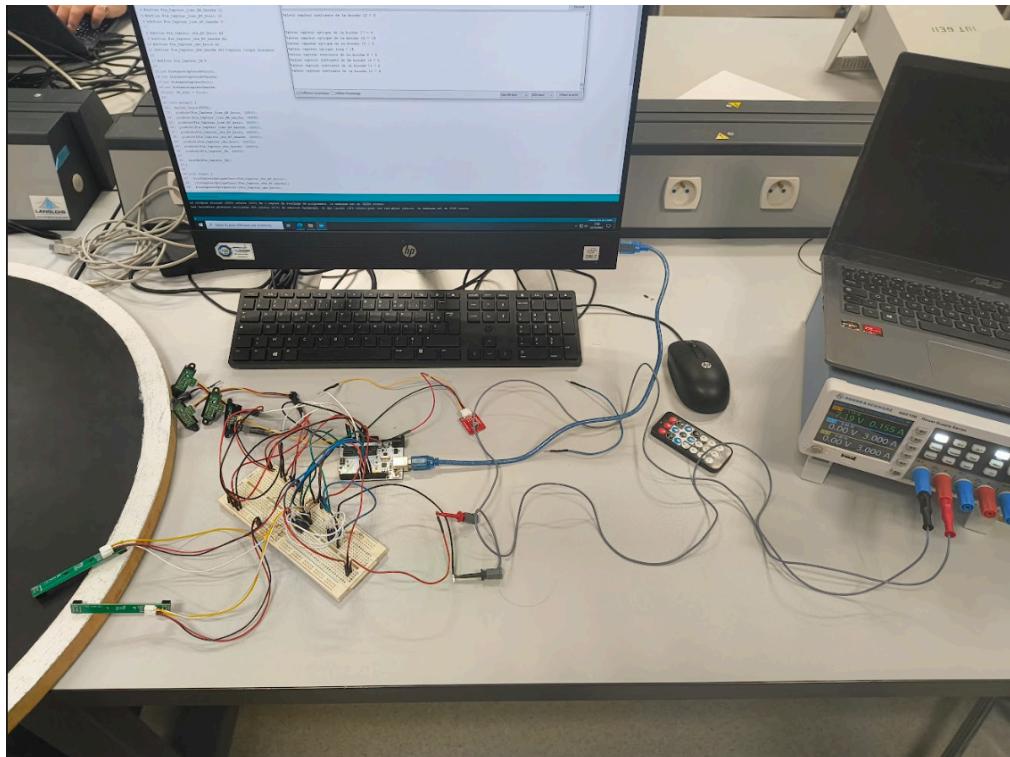
Pour réaliser ce prototypage, nous avons utilisées le matériel suivant :

- Breadboard
- Alimentation de table à 7.2V et une limite de courant à 0.3A
- Arduino Uno
- Ordinateur
- Dohyo
- Télécommande infrarouge

- Composants constituant la majorité du schéma électrique acquisition
- Pinces banane griffes-fils
- Règle

Lors de ce prototypage, nous n'avons pas réalisé les tests avec les différents condensateurs car à ce moment, nous n'avions pas encore évalué leurs valeurs. Le non placement des condensateurs n'a pas un gros impact sur le fonctionnement des composants car ils sont placés pour améliorer la stabilité des composants et non pour assurer leur réel fonctionnement.

A l'aide du matériel énoncés précédemment, nous avons réalisées le prototypage suivant :



**Figure 48 : Prototypage de la partie acquisition du système**

Après avoir réalisé le prototypage suivant, nous devons téléverser le programme dans la carte Arduino et alimenter le montage à partir du régulateur linéaire de l'Arduino. Pour l'utiliser, nous devons connecter l'alimentation de table à la broche V/N de l'Arduino et alimenter la breadboard à partir de la broche +5V de l'Arduino.

Nous rappelons que l'alimentation de table doit avoir comme configuration sur sa sortie une tension de +7.2V pour simuler la tension générée par la batterie et une limite de courant de 0.3A.

Pour mesurer la certitude des capteurs optiques, nous utiliserons une règle pour mesurer la distance entre un mur droit et le capteur optique.

Pour mesurer la certitude des capteurs de contrastes, nous les essayerons 1 par 1 en les faisant visualiser la couleur noire et la couleur blanche du dohyo.

### Résultats attendus :

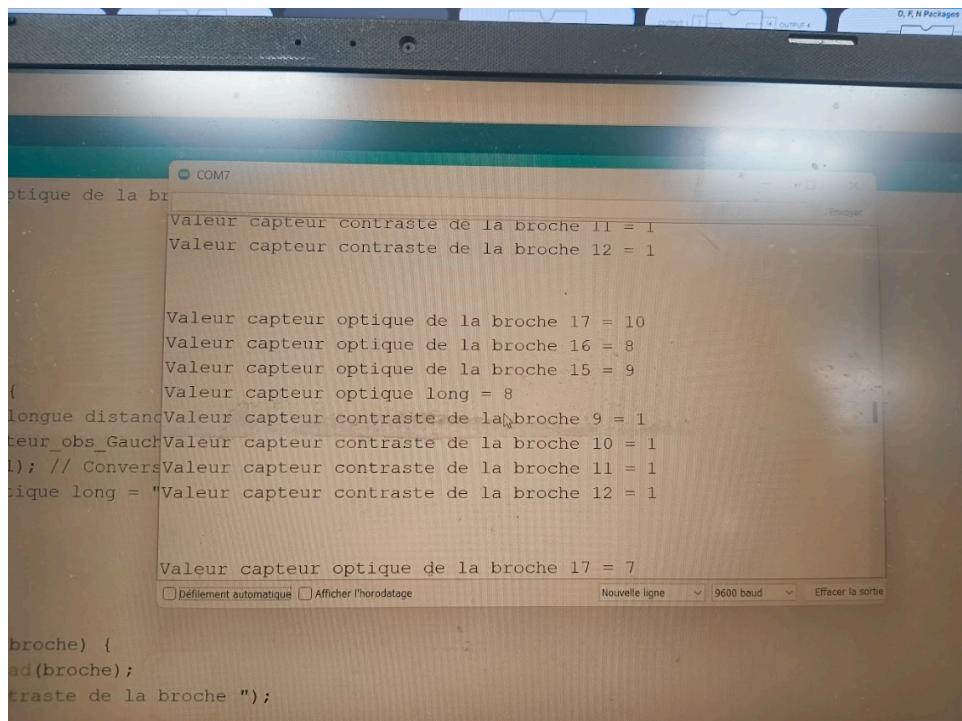
IUT Bordeaux Département GEII	Référence : RMS_DDC_EQ11 Révision : 3 – 04/11/2024	52/56
----------------------------------	---	-------

- Le programme n'envoie rien au moniteur de série tant que nous n'avons pas envoyé de signal infrarouge quelconque sur la capteur infrarouge
- Après la réception d'un quelconque signal infrarouge, le moniteur de série doit nous afficher les résultats suivants :
  - Les 4 distances en cm perçus par les capteurs optiques dans leur gamme de distance de mesure avec une tolérance de +/- 1 cm
    - Capteur optique "court" -> Voit entre 3 et 20 cm
    - Capteur optique "long" -> Voit entre 10 et 80 cm
  - Les 4 valeurs reçues à partir des capteurs de contrastes
    - Si sur noir : 1
    - Si sur blanc : 0

### Résultats obtenus :

Le programme n'envoyait rien tant que nous n'avons pas envoyé un signal infrarouge vers le capteur infrarouge.

Après l'envoi d'un signal infrarouge, les valeurs de chaque capteur optique dans leur gamme de distance étaient conformes à +/- 1 cm et les capteurs de contrastes renvoient bien 1 ou 0 en fonction de la couleur qu'ils voyaient. Comme le prouve l'image ci-dessous :



**Figure 49 : Affichage des différentes valeurs renvoyées par les différents capteurs**

**Statut de l'essai :** Conforme

**Problèmes rencontrés :** La distance renvoyée par le capteur longue distance était pas représentatif de la distance mesurée. Le problème était que nous avions changé

de capteur longue distance et que chaque capteur ne fonctionnait pas totalement de la même façon malgré la même référence. Nous avons donc changé de capteur jusqu'à en avoir un qui nous renvoie les distances qu'on mesure en réalité. Nous avons donc gardé ce capteur et tous les autres du capteur du prototypage pour les utiliser lors de la fabrication de la carte électronique.

## 4.4 Conclusion de la simulation / prototypage rapide du produit

Suite aux différents prototypes et essais, nous pouvons dire que chaque partie est conforme et répond aux attentes indépendamment des autres. Il nous reste à essayer que l'assemblage des différentes parties pour s'assurer du bon fonctionnement de notre robot.

## 5. Conclusion de la conception du produit

Concluez sur la conception du produit vis-à-vis des exigences client en insistant plus particulièrement sur les non-conformités identifiées, les causes possibles et les solutions envisagées.

## 6. Matrice de conformité du produit

Ce chapitre synthétise par l'intermédiaire d'un tableau la conformité du produit développé par rapport aux exigences issues du Cahier des Charges.

Exigence	Méthodes Vérification	Eléments vérifiant l'exigence	Statut
EXIG_CHASSIS_DIMENSIONS			
EXIG_MASSE			
EXIG_INTEGRITE_MECA			
EXIG_FIXATION_CARTE			
EXIG_AUTONOMIE			
EXIG_SECUR_BATT			
EXIG_ADVERSAIRE			
EXIG_LUMINOSITE			
EXIG_DEPART			

Robot Mini-Sumo (RMS)

EXIG_DEPLACEMENT			
EXIG_COMPORTEMENT			
EXIG_CARTE			
EXIG_DELAI			
EXIG_COUT			
EXIG_DRIVE			
EXIG_FORMAT_DOC			
EXIG_NOM_DOC			