

Dossier De Conception (DDC)

du projet

Robot Mini-Sumo

Responsabilité documentaire

Action	NOM Prénom	Fonction	Date	Signature
Rédigé par	LARJUZAN Noah DALU-BEAUTÉ Mathias CHEVALIER Mathieu L'HARIDON Mathis TEMPLIER-BOURDA Tancrède THIERCEAULT Faustine	Technicien	08/10/2024	
Approuvé par	F. GIAMARCHI (IUT GEII Bdx)	Organisateur du tournoi Robot Mini-Sumo	07/10/2024	
Approuvé par	L. THEOLIER (IUT GEII Bdx)	Responsable Robot Mini-Sumo GEII IUT de Bdx	07/10/2024	

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	08/10/2024	Première publication

Documents de références

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

Table des matières

1. Nature du document	5
2. Conception préliminaire du produit	5
2.1 Architecture Électronique	5
2.1.1 Choix des capteurs adversaire	7
2.1.2 Choix des capteurs de sol	9
2.1.3 Choix du récepteur IR	10
2.1.4 Possibilité d'ajouter un multiplexeur	11
2.1.5 Choix du pont en H	12
2.1.6 Choix du régulateur de tension	14
2.1.7 Choix de la batterie	16
2.2 Architecture Informatique	19
2.2.1 Fonctions d'acquisition	21
2.2.2 Fonctions d'action	21
2.2.3 Fonctions d'énergie	23
2.2.4 Fonctions de traitement	23
Fonction SignalNec()	23
Fonction Comportement()	24
2.3 Conclusion de la conception préliminaire du produit	24
3. Conception détaillée du produit	25
3.1 Conception détaillée du robot mini-sumo	25
3.2 Conception détaillée de la partie acquisition	26
3.2.1 Capteurs adversaires	26
3.2.1.1 Schéma électrique des capteurs adversaires	26
3.2.1.2 Code informatique des capteurs adversaires	27
3.2.2 Capteurs de sol (si nécessaire)	28
3.2.2.1 Schéma électrique des capteurs de sol	28
3.2.2.2 Code informatique des capteurs de sol	30
3.2.3 Capteurs de télécommande	31
3.2.3.1 Schéma électrique des capteurs de télécommande	31
3.2.3.2 Code informatique des capteurs de télécommande	32
3.3 Conception détaillée de la partie action	32
3.3.1 Pont en H	32
3.3.1.1 Schéma électrique du pont en H	32
3.3.1.2 Code informatique du pont en H	35
3.4 Conception détaillée de la partie énergie	36
3.4.1 Pont diviseur de tension	36
3.4.1.1 Schéma électrique des ponts diviseurs de tension	37
3.4.2 régulateur de tension	38
3.4.2.1 Schéma électrique du régulateur de tension	38

3.5 Conception détaillée de la partie traitement	40
3.5.1 Le microcontrôleur ATMEGA328P-PU	40
3.4.1.1 Schéma électrique du microcontrôleur	40
3.3.1.2 Code informatique du microcontrôleur	41
4. Dérisquage des solutions techniques retenues	41
4.1 Dérisquage des capteurs de contraste	42
4.2 Dérisquage des capteurs adversaires	44
4.3 Dérisquage des capteurs analogiques	45
4.2 Dérisquage du récepteur démodulateur	46
4.3 Dérisquage du pont en H	47
4.3 Dérisquage de la batterie	49
4.4 Dérisquage du pont diviseur de tension et de l'AOP	50
4.5 Conclusion de la simulation / prototypage rapide du produit	53
5. Conclusion de la conception du produit	53

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_01

Exigences client vérifiées par préconception : EXIG_AUTONOMIE, EXIG_SECUR_BATT, EXIG_ADVERSAIRE, EXIG_LUMINOSITE, EXIG_DEPART, EXIG_DEPLACEMENT, EXIG_COMPORTEMENT

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

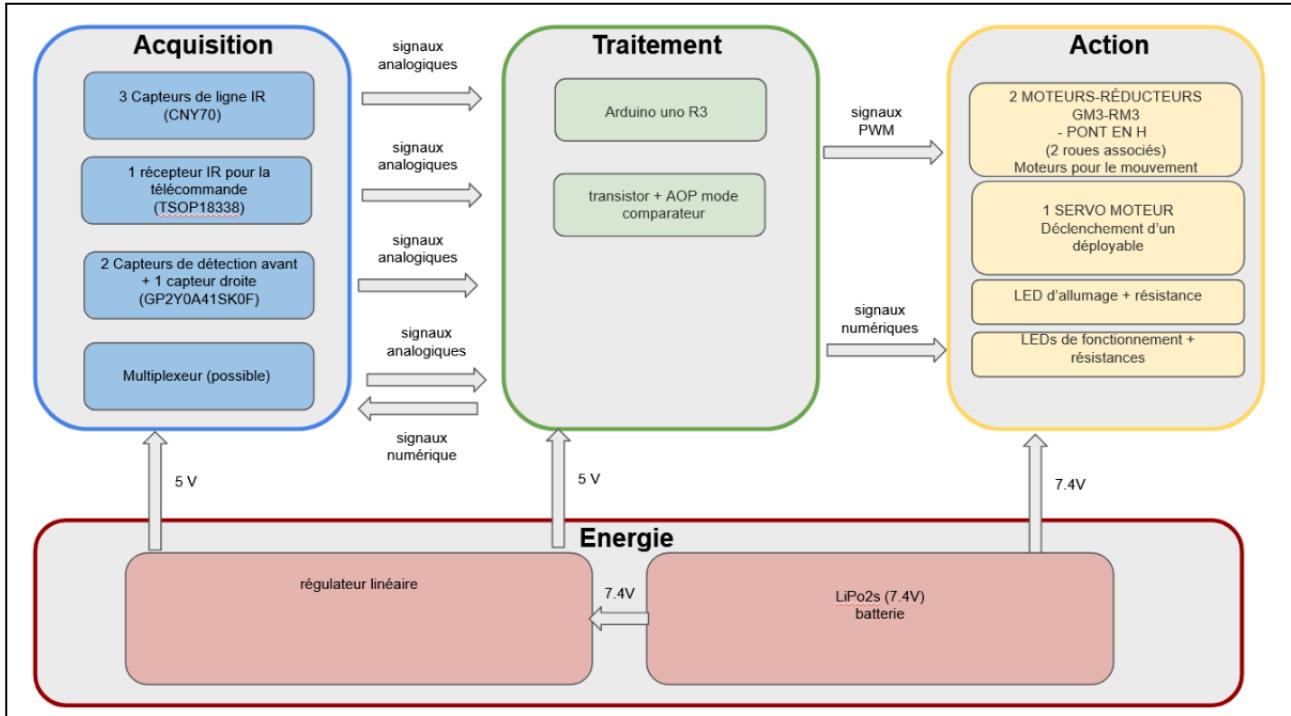


Figure 1 : Architecture électronique de la carte

Pour la partie Acquisition, en accord avec la stratégie de combat et le cahier des charges, nous avons choisi d'utiliser trois capteurs de ligne (capteurs de contraste), trois capteurs d'adversaire et un récepteur IR. Nous avons aussi explorer la possibilité d'ajouter un multiplexeur analogique si nous venions à manquer de broches analogiques sur le microcontrôleur Arduino Uno (détaillez [§2.1.4](#)).

Pour la partie énergie, nous avons dû choisir une batterie adaptée aux besoins de notre robot tout en respectant le cahier des charges. De plus, un régulateur linéaire a été nécessaire pour alimenter certaines parties du robot qui requièrent une tension plus basse. Nous avons détaillé l'ensemble de nos choix dans les sections [2.1.6](#) et [2.1.7](#).

Concernant la partie traitement, il a fallu sélectionner un processeur répondant aux exigences du cahier des charges. Après plusieurs heures de recherche, nous avons opté pour une carte Arduino. En outre, nous avons intégré un transistor et un amplificateur opérationnel (AOP) en mode comparateur pour satisfaire une condition du cahier des charges : protéger la batterie en la coupant avant que la tension ne descende en dessous d'un seuil critique.

Aussi, pour la partie action se compose de trois grandes parties : les déplacements, les LEDs d'affichage et le déployable du robot-sumo.

Nous utilisons les moteurs fournis avec la structure du robot ainsi que leurs moto-réducteurs. Pour les contrôler nous utilisons un circuit pont en H basé sur le composant DRV8833, qui permettra au robot de se déplacer selon ses besoins.

Pour les installations lumineuses du robot, nous avons opté pour une LED verte de fonctionnement qui sera reliée sur la batterie ainsi que d'une LED bleue qui s'allumera lorsque le robot sera en combat.

Pour finir, nous avons opté pour un ServoMoteur et un système mécanique avec un ressort pour installer une rampe déployable sur notre robot.

2.1.1 Choix des capteurs adversaire

Référence de pré-conception : CPR_02

Exigences client vérifiées par préconception : EXIG_ADVERSAIRE

Pour choisir les capteurs d'adversaire, nous avons fait une FAD. Nous avions le choix entre trois capteurs de la marque Sharp : le GP2Y0A02YK, le GP2Y0A021YK et le GP2Y0A41SK0F. Pour les départager, nous les avons comparés sur leur distance maximale de détection, leur facilité d'utilisation du signal de sortie, leur compatibilité de la tension avec les sources disponibles et leur consommation en courant.

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 max de détection	Facilité d'utilisation du signal de sortie	Faible consommation en courant			
Capteur de mesure Sharp GP2Y0A41SK0F	1	30 cm	x	-0.3/7+	12/22	625	Prototypage fait
		5	5	5	5		
Capteur de mesure Sharp GP2Y0A021YK	0	80 cm	x	-0.3/7+	30/40	375	
		5	5	5	3		
Capteur de mesure Sharp GP2Y0A02YK	0	150 cm	x	-0.3/7+	33/50	375	
		5	5	5	3		

Figure 2 : FAD pour le choix des capteurs de détection d'ennemis

La FAD nous a montré que le capteur GP2Y0A41SK0F est le plus apte à répondre aux contraintes du projet. Pour répondre pleinement à la stratégie de combat de notre groupe, nous avons choisi d'en placer deux sur la face avant et un sur la face de droite (Figure 3). Nous allons alors avoir besoin de trois broches analogiques pour récupérer les données de ces trois capteurs.

Robot Mini-Sumo (RMS)

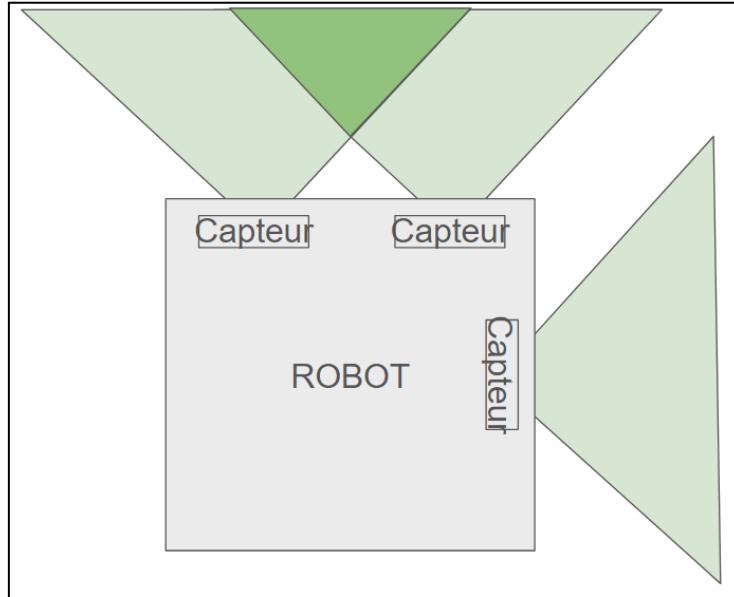
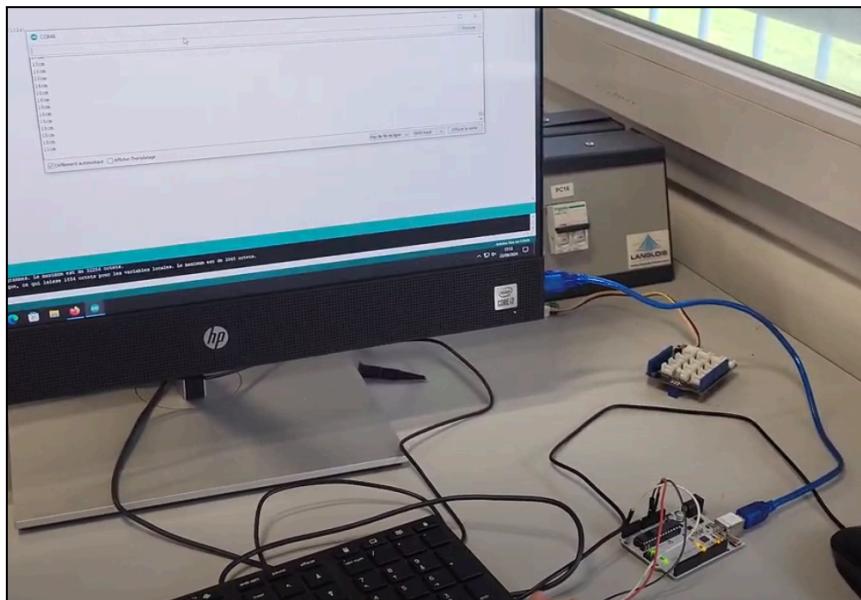


Figure 3 : Schéma schématique des capteurs d'adversaire

Nous avons confirmé notre choix par prototypage rapide. Nous avons branché à un Arduino Uno nos trois capteurs avec leurs broches Vcc sur la broche 5V de l'Arduino, les broches GND sur une des broches GND de l'Arduino et les broches Data des capteurs sur une broche analogique chacun, A0, A1 et A2. Ensuite, nous avons téléchargé un code arduino qui lit les valeurs des broches analogiques rattachées à nos capteurs et nous les transmet via le moniteur série et les interprète en une distance. Nous avons obtenu les résultats suivants :



Prototypage_Cap_Adversaire.mp4

Figure 4 : Prototypage rapide des capteurs d'adversaire

D'après la FAD et le prototypage rapide, nous avons retenu la combinaison de trois capteurs GP2Y0A41SK0F de la marque Sharp (deux devant et un à droite) pour la détection de l'adversaire.

2.1.2 Choix des capteurs de sol

Référence de pré-conception : CPR_03

Exigences client vérifiées par préconception : EXIG_LUMINOSITE

Pour choisir la disposition de nos capteurs de sol CNY70 (capteur de contraste), nous avons fait une FAD. Pour les départager, nous les avons comparés sur leur distance minimale et maximale de détection, leur respect de la stratégie de combat, leur prix et leur disponibilité.

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)				Commentaires (si nécessaire)
		Distance min et max de détection	Respect de la stratégie de combat	Faible Prix	Disponibilité	
1 CNY avant + 0 CNY arrière	0	0 à 5mm	x	1.39€	x	0
		5	0	0	5	
2 CNY avant + 0 CNY arrière	0	0 à 5mm	x	~1.20€	x	0
		5	0	1	5	
0 CNY avant + 1 CNY arrière	0	0 à 5mm	x	1.39€	x	0
		5	0	0	5	
1 CNY avant + 1 CNY arrière	0	0 à 5mm	x	~1.20€	x	0
		5	0	1	5	
2 CNY avant + 1 CNY arrière	1	0 à 5mm	x	~1.10€	x	250
		5	5	2	5	
2 CNY avant + 2 CNY arrière	0	0 à 5mm	x	~1.05€	x	225
		5	3	3	5	

Figure 5 : FAD pour le choix des capteurs de sol

Pour utiliser correctement le capteur de contraste CNY70, nous devons ajouter deux résistances pour chaque capteur.

2.1.3 Choix du récepteur IR

Référence de pré-conception : CPR_04

Exigences client vérifiées par préconception : EXIG_COMPORTEMENT

Grâce à la FAD du récepteur IR, nous avons choisi le récepteur TSOP 18338. Nous avons comparé la facilité de branchement, le respect de la stratégie de combat, la fréquence de réception et la compatibilité de la tension avec les sources disponibles. Malgré le bon résultat obtenu de la carte IR ST027, nous n'avons pas l'autorisation de l'utiliser parce que nous sommes en ESE, par conséquent nous devons créer notre propre alternative.

FICHE D'AIDE A LA DECISION (FAD)							
Solution technique proposée	Conformité à toutes les exigences (oui : 1 / non : 0)	Critères de sélection (valeur de 1 à 5)			Compatibilité de la tension avec les sources disponibles	Total (produit des valeurs précédentes)	Commentaires (si nécessaire)
		Facilité de branchement	Respect de la stratégie de combat	Fréquence (38 kHz)			
IR ST027	0	3 broches	5	38 kHz	5V	625	
TSOP 18338	1	3 broches + 1 condensateur + 1 résistance	3	38 kHz	2,5 à 5,5V	375	Librairie IRemote pour l'utiliser Attention Utilis
IUT de Bordeaux Département GEII	Référence : PRJ_FAD Révision : 1.6 – 19/09/2022						2 / 7

Figure 6 : FAD pour le choix du récepteur Infrarouge

Pour le bon fonctionnement de récepteur IR TSOP 18338, nous devons ajouter une résistance et un condensateur comme recommandé par la datasheet.

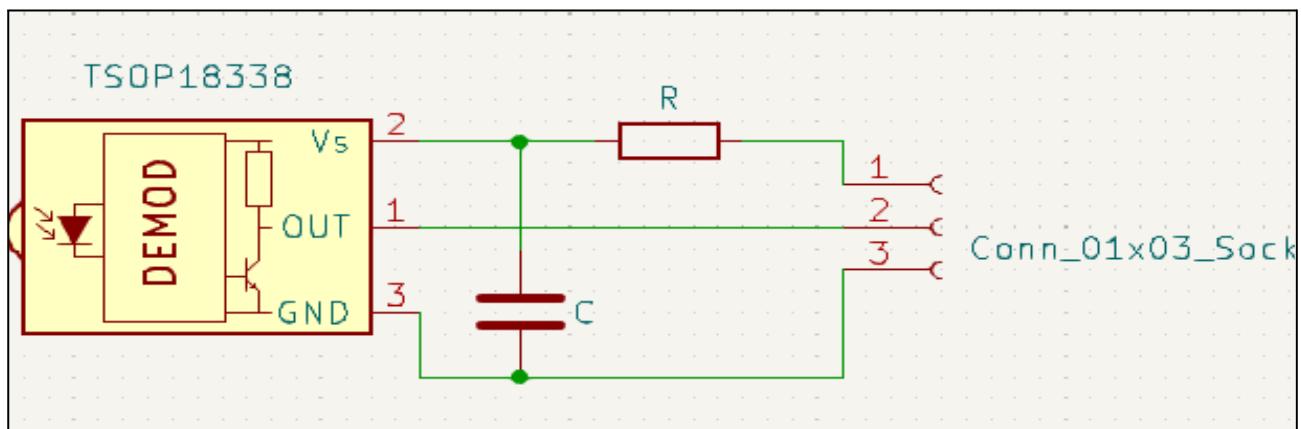


Figure 7 : Schéma du montage associé au récepteur Infrarouge

2.1.4 Possibilité d'ajouter un multiplexeur

Référence de pré-conception : CPR_05

Exigences client vérifiées par préconception : EXIG_ADVERSAIRE

Arduino Uno pour acquérir les valeurs analogiques de nos capteurs. Nous avons donc exploré la possibilité d'utiliser un multiplexeur analogique pour avoir plus de broches analogiques. Avec la carte Arduino Uno, nous sommes restreint à six broches analogiques de A0 à A5. Or nous voulons que nos six capteurs et le contrôle de tension de batterie soient et utilisent des broches analogiques. Nous avons fait un prototypage rapide de cet étage (figure 8).

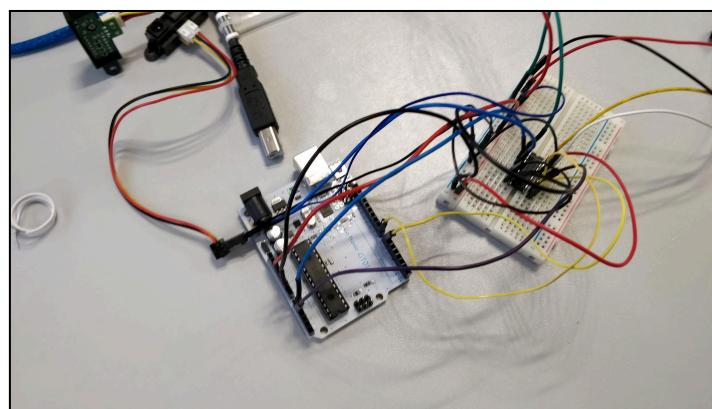


Figure 8 : Prototypage rapide de l'utilisation d'un multiplexeur analogique

Nous réfléchissons encore aux réels plus valus de cet étage.

2.1.5 Choix du pont en H

Référence de pré-conception : CPR_06

Exigences client vérifiées par préconception : EXIG_DEPLACEMENT

Un pont en H est nécessaire afin de répondre à l'exigence de déplacement qui impose la liberté des déplacements du robot SUMO (capable d'avancer, de tourner à gauche ou à droite). En effet, il permet un contrôle bidirectionnel des moteurs. De plus, le pont en H permet d'avoir une indépendance dans le contrôle de chaque moteur (en inversant la polarité des moteurs pour faire pivoter le robot). Enfin, il assure une meilleure gestion énergétique et protège les moteurs contre les surcharges.

Ainsi, la [Fiche d'Aide à la décision](#) de l'exigence déplacement nous a permis de déterminer et sélectionner le pont en H de façon judicieuse en validant plusieurs critères afin de les différencier et de choisir le pont en H le plus adapté à notre besoin, c'est à dire alimenter les deux moteurs des deux roues du robot SUMO.

Ainsi les critères de sélection sont :

- Compatibilité de la tension avec les sources disponibles
- Courant max de sortie du pont
- Manière de piloter le pont
- Faible Prix
- Disponibilité

Pont en H disponible :

- [Pololu Commande de 2 moteurs CC DRV8833 2x1,2A](#)
- [Pololu Commande de 2 moteurs CC DRV8835 2x1,2A](#)

Robot Mini-Sumo (RMS)

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)					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		
Pololu Commande de 2 moteurs CC DRV8835 2x1, 2A	0	0 V - 11 V	1.5 A	(Arduino) ou PHASE/E	4,03 €	oui	CONSO 2A ET PONT H	
		5	0	3	4	5		
Pololu Commande de 2 moteurs CC DRV8833 2x1, 2A	1	2.7 V - 10.8 V	2 A	PWM Arduino	5,40€ TTC	oui	4.5/5	
		5	5	4	3	5		
						1500		
						0.00		

Figure 9 : FAD pour le choix du pont en H

Grâce à l'analyse des datasheet des deux ponts en H disponibles, nous avons admis que les tensions sont compatibles avec les sources dans les deux cas. C'est à dire, les sources disponibles du robot sont fixées à 7.4V, ainsi :

- DRV8835 : 0V / 11V → compatible
- DRV8833 : 2.7V / 10.8V → compatible

Pour continuer, afin de déterminer le courant maximal nécessaire pour faire fonctionner les moteurs, nous avons expérimenté en testant à l'aide d'un générateur de tension, du robot sumo ainsi que d'un ampèremètre. Ce qui nous a permis de démontrer le courant maximal consommé par les moteurs en sortie du pont en H.

Courant maximal consommé par les moteur : 1.6 A

Courant I_{max} de sortie :

- DRV8835 : 1.5 A → incompatible
- DRV8833 : 2 A → compatible

De plus, sur les sites marchands nous avons pu vérifier leurs disponibilités ainsi que leurs prix.

Nous avons donc sélectionné avec un score total de 1500 contre 0 le Pololu Commande de 2 moteurs CC DRV8833 2x1,2A car c'est le seul capable de répondre à la consommation maximale nécessaire des moteurs de 1.6 A.

Robot Mini-Sumo (RMS)

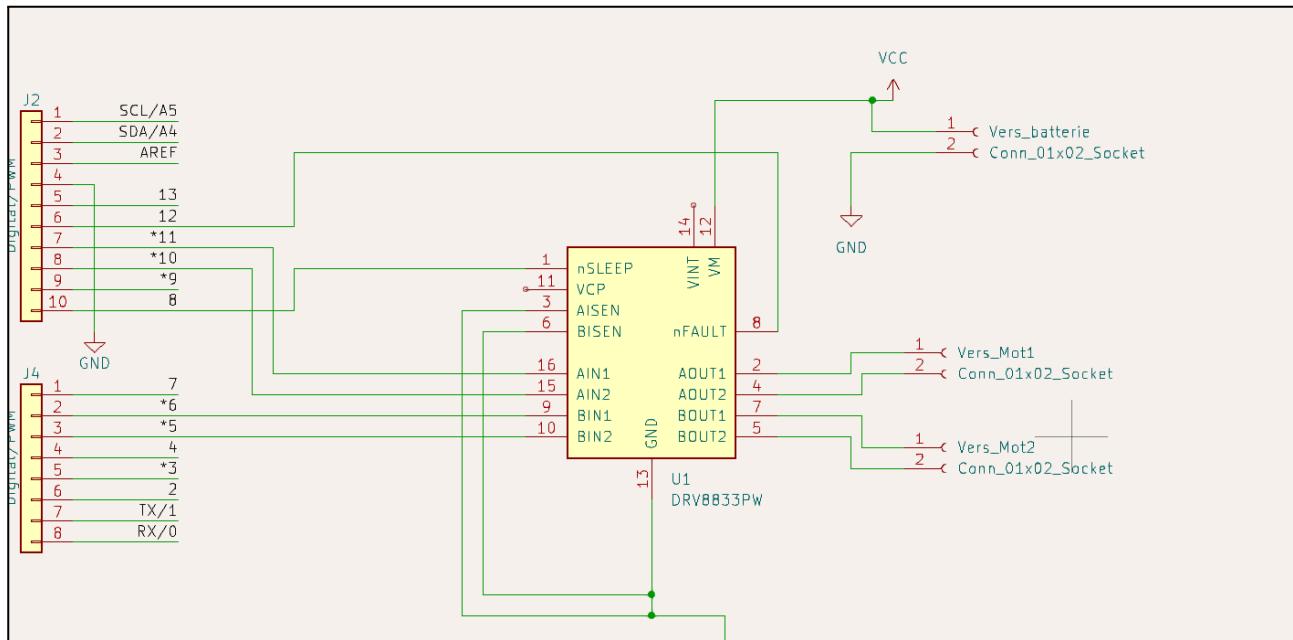


Figure 10 : Schémas préliminaire du montage du pont en H

2.1.6 Choix du régulateur de tension

Référence de pré-conception : CPR_07

Exigences client vérifiées par préconception : EXIG_COMPORTEMENT, EXIG_ADVERSAIRE, EXIG_LUMINOSITE, EXIG_DEPART

Pour la fiche d'aide à la décision concernant le choix du régulateur linéaire, nous avons opté pour le régulateur du type MC33269DT-5.0G IC. Ce choix s'ajuste parfaitement aux spécifications de notre robot sumo et à notre cahier des charges. En effet, le régulateur permet de répondre à un certain nombre d'exigences en termes de traitement et d'acquisition. Il permet d'abaisser la tension à 5V pour alimenter les capteurs et la carte arduino, contrairement à l'alimentation du moteur qui a besoin de 7.4V soit la batterie elle-même.

Pour déterminer ce choix, nous avons comparé 4 types de régulateurs différents, puis nous l'avons dimensionné, testé et validé.

Robot Mini-Sumo (RMS)

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)
		Tension max d'entrée	Tension de sortie	Faible chute de tension sortie-entrée	Courant de sortie max	Boîtier composant facile à utiliser/soudre	Faible Prix	Disponibilité	
LP2985AIM5-3.3/NOPB 3.3V CMS	0	16 V	3.3 V	280 mV	150 mA	2.9mm/1.6mm 5 pins	0.936	90 018	0
		5	0	5	5	5	3	5	
LP2985AIM5-5.0/NOPB 5V CMS	0	16 V	5 V	280 mV	150 mA	2.9mm/1.6mm 5 pins	0.936	5 632	0
		5	5	5	0	4	3	4	
MC33269DT-3.3G IC	0	20 V	3.3 V	1 V	800 mA	5mm/4mm 3 pins	1	2 386	0
		5	0	1	5	3	2	3	
MC33269DT-5.0G IC	1	20 V	5 V	1 V	800 mA	5mm/4mm 3 pins	1	12 584	2250
		5	5	1	5	3	2	3	

Figure 11 : FAD pour le choix du régulateur de tension

Les deux régulateurs fournissant que 3.3V sont incapables d'alimenter les capteurs et la carte arduino. Car ces derniers ont besoin d'une alimentation de 5V. Ils nous restent donc que 2 choix possible :

Pour choisir entre les deux régulateurs de tension restants, nous avons dimensionné celui qui correspondait le mieux aux besoins. Nous avons d'abord évalué la consommation de courant requise en additionnant les différents courants consommés par les éléments qui seront alimenté par le régulateur (+5V) :

- courant de la carte Arduino = 50mA
- courant des 6 capteurs = $3 \times 12 + 3 \times 20 = 96$ mA
- courant récepteur infrarouge = 4.7 mA

$$I_{total} = I_{capteurs} + I_{récepteur\ IR} + I_{carte\ arduino}$$

$$I_{total} = 50 + 96 + 4.7) \times 10^{-3} = 0.1507 A$$

Nous avons besoin d'un courant minimum de 0.150 A pour un bon fonctionnement. De plus, si nous prenons la valeur efficace de ce résultat pour une marge de sécurité, nous obtenons :

$$I_{efficace} = 150 * \sqrt{2} = 212.13 mA$$

Par conséquent, le régulateur le plus adéquat au bon fonctionnement de notre robot est le MC33269DT-5.0G IC. Il délivre assez de courant 800mA contrairement au LP2985AIM5-5.0/NOPB 5V CMS qui lui délivre 150 mA, ce qui est trop juste pour notre robot.

2.1.7 Choix de la batterie

Référence de pré-conception : CPR_08

Exigences client vérifiées par préconception : EXIG_AUTONOMIE

Pour la fiche d'aide à la décision concernant le choix de la batterie, nous avons opté pour la batterie Lipo CONRAD ENERGY 7.4 V 800 mAh. Ce choix s'ajuste parfaitement aux spécifications de notre robot sumo et à notre cahier des charges énergétique. En effet, l'exigence en termes d'autonomie est adaptée à la taille du robot et au temps de fonctionnement minimum requis.

Nous avions besoin d'une batterie aux dimensions spécifiques, car elle doit s'insérer dans un emplacement dédié prévu à cet effet, qui est de 57mm x 31mm x 17mm. Concernant le temps de fonctionnement, le cahier des charges impose que l'énergie stockée permette un minimum de 15 minutes de fonctionnement pour un combat, soit 90 min en fonctionnement à vide. Grâce à ces exigences, nous avons pu dimensionner la batterie de manière précise.

Nous avions le choix entre trois batteries, comme illustré dans le tableau, chacune offrant un courant minimal d'un ampère circulant pendant une durée différente.

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	Faible Prix	Disponibilité		
Lipo CONRAD ENERGY 7.4 V 1200 mAh	0	72x36x12 0	21.99 1	20 2	0	trop grande mais bonne charge
Lipo CONRAD ENERGY 7.4 V 800 mAh	1	56x31x17 5	9.99 4	39 3	60	bonne taille et bonne charge
Lipo CONRAD ENERGY 7.4 V 450 mAh	0	56x31x11.5 1	9.49 4	0 0	0	bonne taille mais pas assez de charge
IUT de Bordeaux Département GEII	Référence : SUMO_FAD Révision : 1.6 – 23/09/2024					7 / 7

Figure 12 : FAD pour le choix de la batterie

Pour choisir la batterie adéquate, j'ai dimensionné celle qui correspondait le mieux aux besoins. J'ai d'abord évalué la consommation de courant requise en additionnant les différents courants consommés par les éléments de la partie action :

- courant led bleue = 3.6 mA

- courant led verte = 1.1 mA
- courant de la carte Arduino = 50mA
- courant des 6 capteurs = $3*12+3*20 = 96$ mA
- courant récepteur infrarouge = 4.7 mA
- courant des deux moteurs à vide = 170 mA

$$I_{total} = I_{led bleue} + I_{led verte} + I_{capteurs} + I_{récepteur IR} + I_{carte arduino} + I_{moteurs}$$

$$I_{total} = (1.1 + 3.6 + 50 + 96 + 4.7 + 170) * 10^{-3} = 0.324 A$$

Ensuite, afin de calculer les ampères-heures, j'ai déterminé le temps d'utilisation en heures:

$$t = 90 \text{ min} = 1.30 \text{ h}$$

J'ai ensuite multiplié ces deux valeurs pour obtenir la capacité en ampères-heures :

$$B = I_{total} * t = 0.324 * 1.3 = 0.421 Ah = 421 mAh$$

Pour avoir une marge de sécurité, on calcule la valeur efficace :

$$B_{efficace} = 421 * \sqrt{2} = 595 mAh$$

Il apparaît que la batterie Lipo CONRAD ENERGY 7.4 V 450 mAh ne permet pas à notre robot de fonctionner durant le temps requis. Nous devons donc choisir entre les deux autres options restantes : la Lipo CONRAD ENERGY 7.4 V 1200 mAh et la Lipo CONRAD ENERGY 7.4 V 800 mAh.

De plus, grâce à la fiche technique (datasheet), nous disposons des dimensions de ces batteries. Il est important de noter que :

- Lipo CONRAD ENERGY 7.4 V 1200 mAh mesure 72 mm x 36 mm x 12 mm
- Lipo CONRAD ENERGY 7.4 V 800 mAh mesure 56 mm x 31 mm x 17 mm

Selon les spécifications du cahier des charges, l'emplacement réservé à la batterie est de 57 mm x 31 mm x 17 mm. Par conséquent, seule la batterie Lipo CONRAD ENERGY 7.4 V 800 mAh correspond à ces dimensions et peut être retenue.

De plus, nous vérifions si la batterie est toujours acceptable avec la marge de sécurité. Pour cela, nous enlevons 20 % de la valeur initiale. En effet, la batterie ne doit pas descendre en dessous de ces 20% de batterie maximum. Soit :

$$800 * 0.2 = 160 mAh$$

$$800 - 160 = 640 mAh$$

$$640 mAh > 595 mAh$$

La batterie reste au-dessus de notre consommation demandée.

Robot Mini-Sumo (RMS)

En conclusion, après avoir évalué les besoins en courant et les dimensions disponibles, la batterie Lipo CONRAD ENERGY 7.4 V 800 mAh s'avère être la meilleure option pour garantir le bon fonctionnement du robot tout en respectant les contraintes d'encombrement

2.2 Architecture Informatique

Référence de pré-conception : CPR_09

Exigences client vérifiées par préconception : EXIG_ADVERSAIRE, EXIG_LUMINOSITE, EXIG_DEPART, EXIG_DEPLACEMENT, EXIG_COMPORTEMENT, EXIG_SECUR_BATT

Ci-dessous vous trouverez la synoptique informatique préliminaire décrivent les fonctions informatiques que nous utilisons afin de remplir les exigences clients.

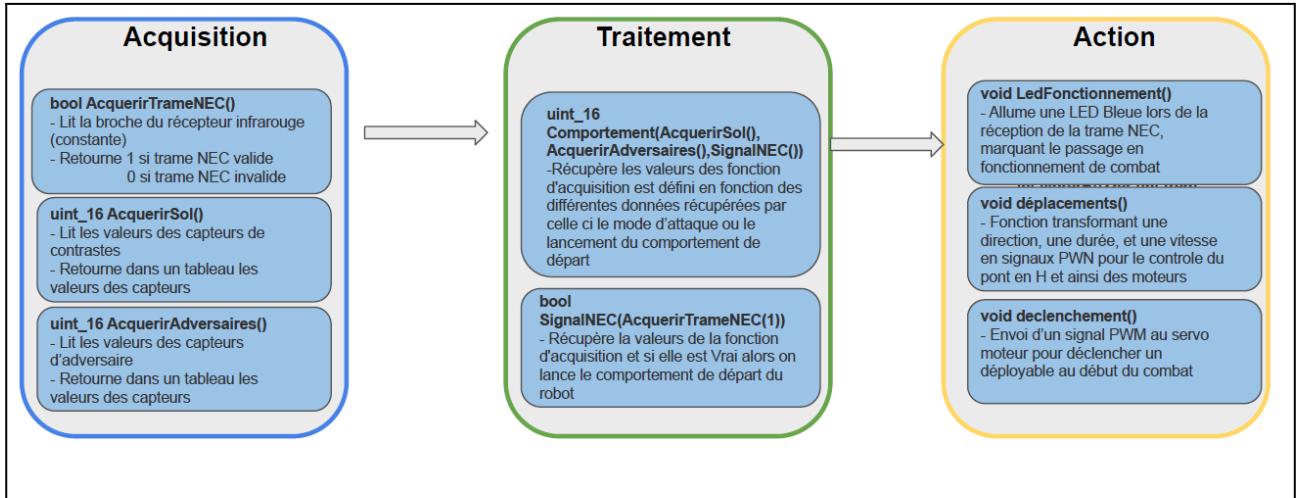


Figure 13 : Synoptique préliminaire informatique

Robot Mini-Sumo (RMS)

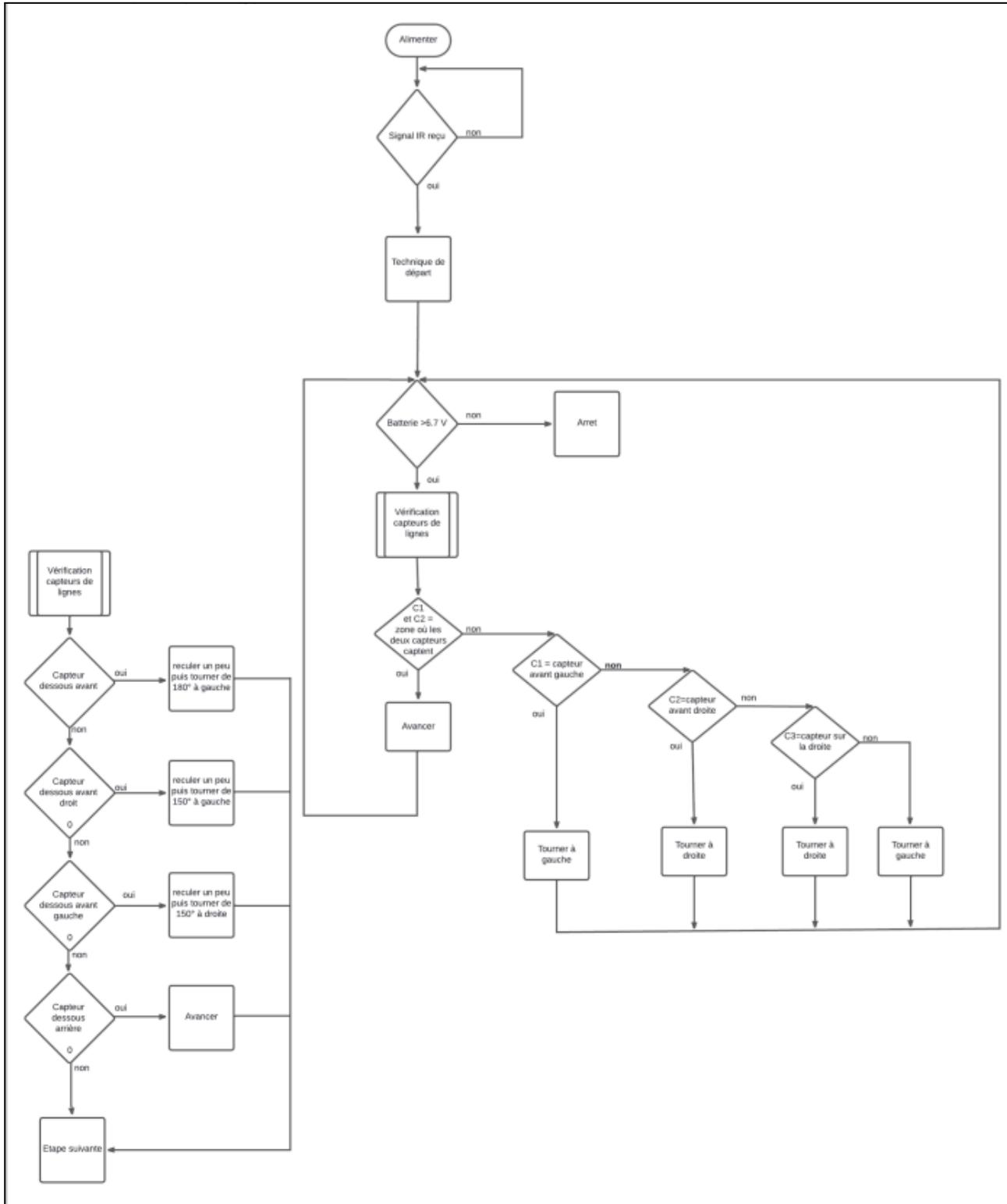


Figure 14 : Logigramme correspond au fonctionnement au robot

2.2.1 Fonctions d'acquisition

Référence de pré-conception : CPR_10

Exigences client vérifiées par préconception : EXIG_ADVERSAIRE, EXIG_LUMINOSITE, EXIG_DEPART

Nous avons choisi de faire trois fonctions informatiques pour acquérir les valeurs des différents capteurs. La première fonction, AcquerirTrameNEC(), permet de lire le récepteur IR et de vérifier que la trame reçue est bien une trame NEC provenant de la télécommande IRC01. Elle renvoie un booléen (1 = Trame NEC valide, 0 = Trame NEC invalide). Les fonctions deux et trois, AcquerirSol() et AcquerirAdversaires(), permet de lire les valeurs des capteurs de contrastes et des capteurs d'adversaire respectivement et les renvoie dans un tableau.



Figure 15 : Synoptique préliminaire informatique de la partie Acquisition

2.2.2 Fonctions d'action

Référence de pré-conception : CPR_11

Exigences client vérifiées par préconception : EXIG_DEPLACEMENT,

Pour le fonctionnement de notre robot, nous sommes partis sur une base de 3 fonctions. La première fonction consiste juste en l'allumage d'une LED lors de la réception de la trame NEC, indiquant le passage du robot dans une phase de combat.

Notre seconde fonction contrôle les déplacements du robot comme stipulé dans le cahier des charges. En injectant une direction, une vitesse ainsi qu'une durée de mouvement, le robot est libre de se déplacer en suivant les besoins de la partie traitement.

Cette fonction consiste en une transformation d'un besoin en signal PWM envoyé vers le pont en H pour contrôler les moteurs.

Notre dernière fonction Action consiste au déclenchement d'un Servomoteur relié à un déployable qui sera installé sur notre robot. Le contrôle du Servo-moteur sera effectué à l'aide d'un signal PWM généré par l'ATMEGA 328p.

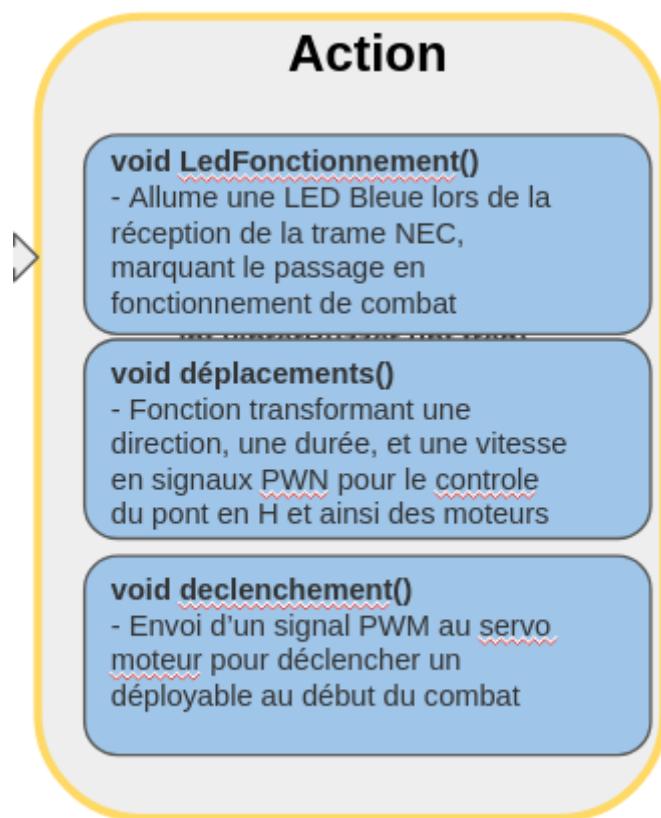


Figure 16 : Synoptique préliminaire informatique de la partie Action

2.2.3 Fonctions d'énergie

Référence de pré-conception : CPR_12

Exigences client vérifiées par préconception :

Les exigences liées à la partie énergétique seront résolues sans fonction informatique juste avec l'électronique analogique et non informatique.

2.2.4 Fonctions de traitement

Référence de pré-conception : CPR_13

Exigences client vérifiées par préconception : EXIG_DEPLACEMENT,
EXIG_COMPORTEMENT,

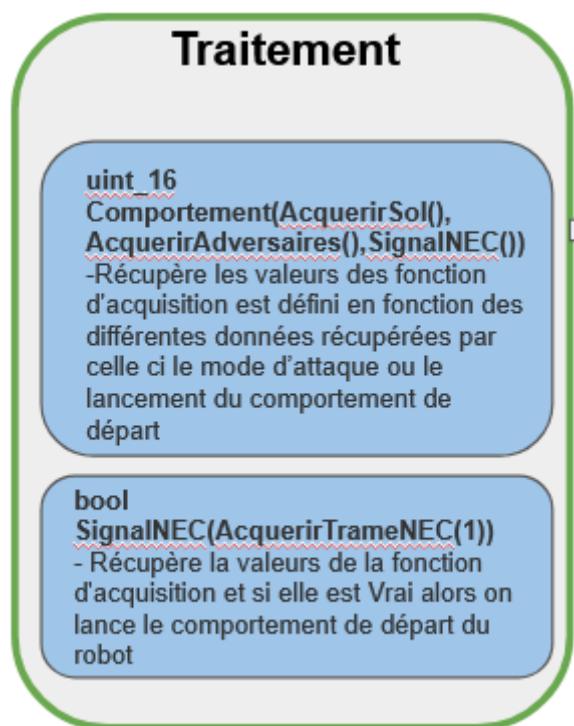


Figure 17 : Synoptique préliminaire informatique de la partie Traitement

Fonction **SignalNec()**

La fonction **SignalNec()** a pour objectif de vérifier la validité de la trame NEC reçue. Elle analyse la trame pour déterminer si elle correspond à celle attendue. Si la trame est correcte, la fonction retourne une valeur booléenne **true**, sinon elle retourne **false**.

Fonction **Comportement()**

La fonction **Comportement()** exploite les données validées par la fonction **SignalNec()**, ainsi que d'autres informations captées par les capteurs du système. En fonction de ces valeurs, elle détermine l'un des quatre comportements suivants :

- **Stratégie de départ** : Mise en place de l'algorithme initial pour définir le point de départ et la stratégie globale.
- **Recherche de l'adversaire** : Identification de la position de l'adversaire, en fonction des informations recueillies par les capteurs.
- **Attaque** : Passage à une stratégie offensive lorsque l'adversaire est détecté.
- **Éloignement des lignes** : Lorsque des lignes sont détectées (telles que des limites de terrain ou des marquages au sol), le système active un comportement d'évitement pour éloigner le robot des bordures.

Ces quatre comportements permettent au système de réagir de manière dynamique et adaptée aux conditions environnantes, assurant ainsi son bon fonctionnement dans des scénarios variés.

2.3 Conclusion de la conception préliminaire du produit

Pour l'ensemble des parties (Acquisition, Traitement, Action et Énergie), nous avons réussi à trouver des solutions techniques permettant de répondre aux exigences du cahier des charges. Nous avons déjà retenu des étages électroniques qui assureront un bon fonctionnement du robot mini sumo. Nous sommes prêts à passer à l'étape suivante qui est la 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

Chaque bloc fonctionnel doit faire l'objet d'un chapitre de conception détaillé, présenté comme suit.

Référence de conception : CDT01

Exigences client vérifiées : EXIG_AUTONOMIE, EXIG_SECUR_BATT, EXIG_ADVERSAIRE, EXIG_LUMINOSITE, EXIG_DEPART, EXIG_DEPLACEMENT, EXIG_COMPORTEMENT, EXIG_CARTE,

A la suite de la conception préliminaire, l'activité de conception détaillée a été menée. Le schéma électrique complet de la carte est fourni ci-dessous ainsi que le synoptique électronique détaillé.

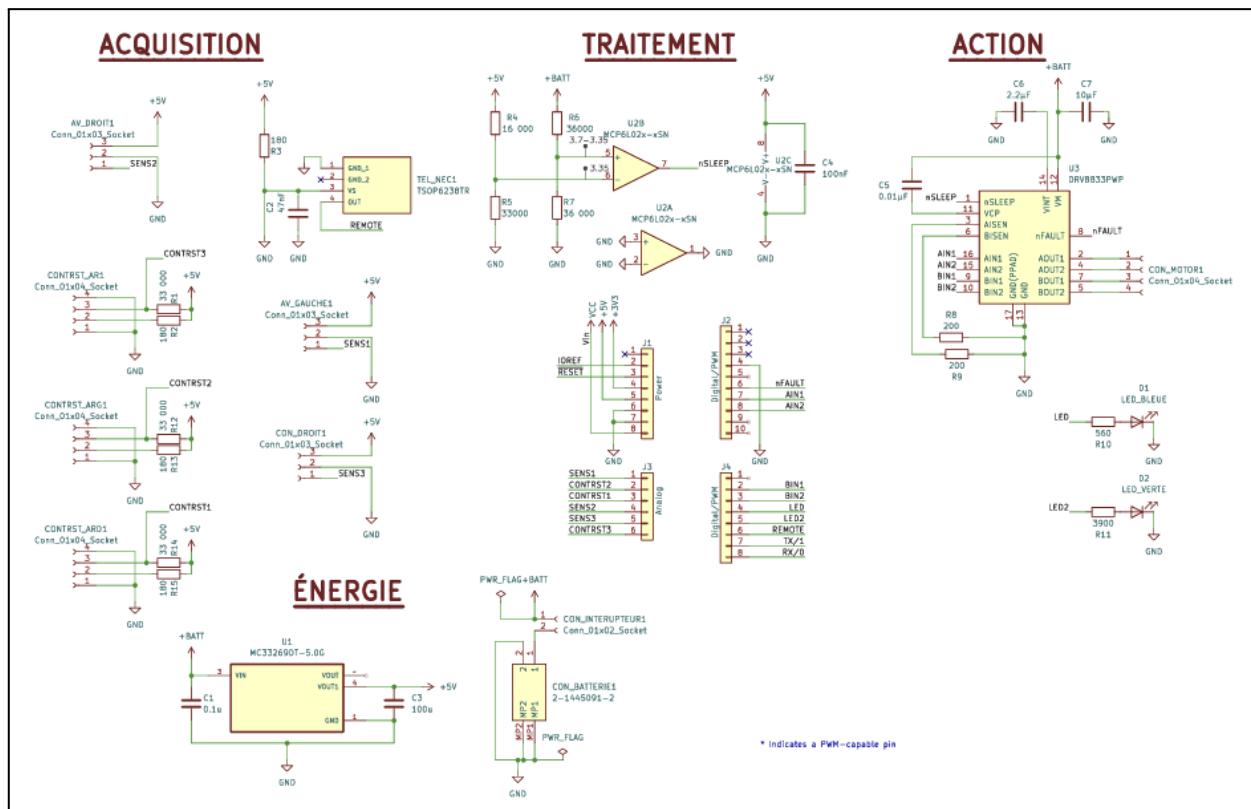


Figure 18 : Schéma électrique de la carte robot mini-sumo

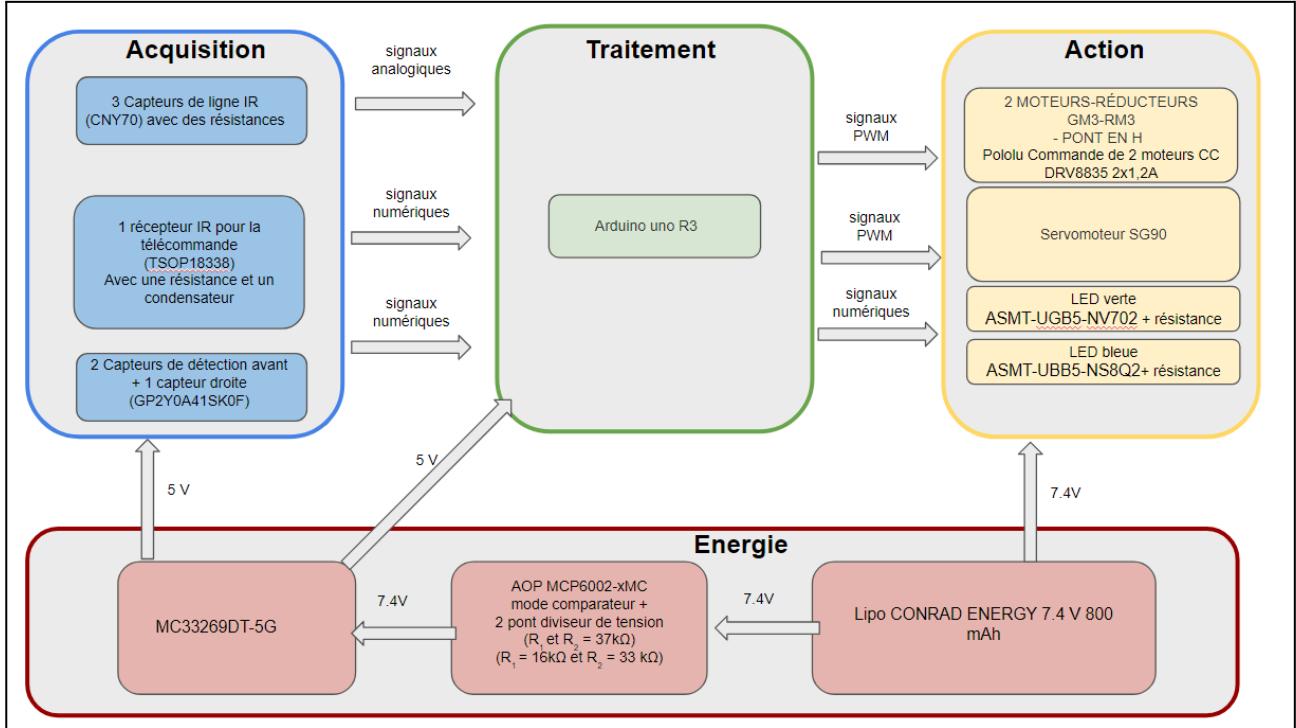


Figure 19 : Synoptique électronique détaillé du projet

3.2 Conception détaillée de la partie acquisition

3.2.1 Capteurs adversaires

Référence de conception : CDT02

Exigences client vérifiées : EXIG_ADVERSAIRE

3.2.1.1 Schéma électrique des capteurs adversaires

Comme mentionné en conception préliminaire, nous allons utiliser le capteur GP2Y0A41SK0F. Nous allons en avoir trois sur le robot sumo (deux devant et un à droite) (Figure 20)

Robot Mini-Sumo (RMS)

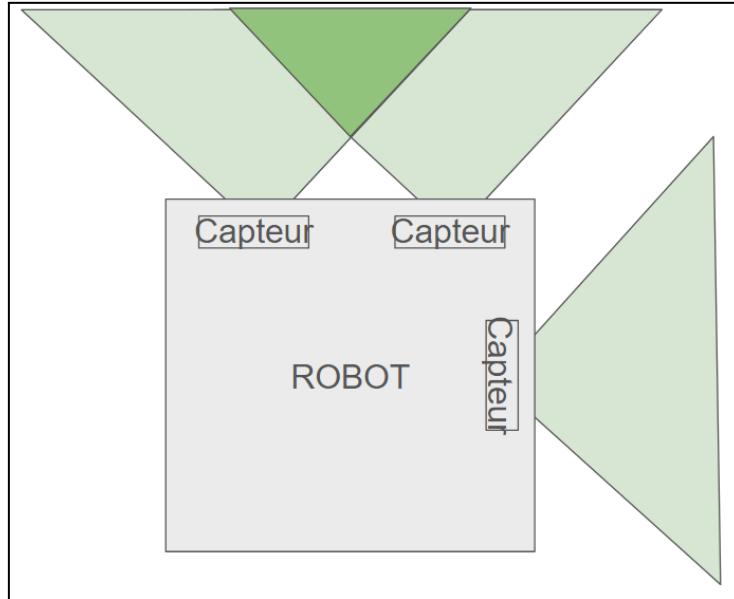


Figure 20 : Schéma schématique des capteurs d'adversaire (Vue de haut)

Pour les relier au microcontrôleur, nous allons ajouter au shield trois connecteurs de trois broches. La broche 1 sera les données du capteur directement reliée à une broche analogique du microcontrôleur, la broche 2 sera la masse (GND) et la broche 3 sera la tension d'alimentation (Vcc = +5V) (Figure 21).

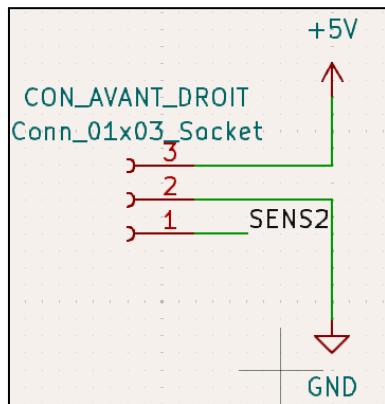


Figure 21: Schéma électrique d'un capteur d'adversaire sur le shield

Nous avons fait très attention au sens des connecteurs et à leurs positionnements lors du routage de cette partie.

3.2.1.2 Code informatique des capteurs adversaires

La fonction AcquerirAdversaires met à jour 3 pointeurs et émet ses données sur le port série pour le débogage.

Elle fonctionne en deux temps, premièrement nous convertissons le chiffre que nous donne l'adc en volts puis nous convertissons cette tension en distance.

```
void AcquerirAdversaires(float *DistanceCapteurAD,float*DistanceCapteurAG,float  
*DistanceCapteurD ) {  
    // ne pas appeler plus de 50 fois par seconde  
  
    // conversion de la tension en chiffre digital  
    float TensionCapteurAD =  analogRead(data_pinAD) * 0.0048828125;  
    float TensionCapteurAG =  analogRead(data_pinAG) * 0.0048828125;  
    float TensionCapteurD =  analogRead(data_pinD) * 0.0048828125;  
    // conversion des chiffre digital en distance  
    if (13 * pow(TensionCapteurAD, -1) <= 30){  
        *DistanceCapteurAD = 13 * pow(TensionCapteurAD, -1); // mise à jour du pointeur  
    }  
    else{  
        // valeur d'erreur si la distance est invalide  
        *DistanceCapteurAD = -1;  
    }  
  
    if (13 * pow(TensionCapteurAG, -1) <= 30){  
        *DistanceCapteurAG = 13 * pow(TensionCapteurAG, -1);  
    }  
    else{  
        *DistanceCapteurAG = -1;  
    }  
    if ( 13 * pow(TensionCapteurD, -1) <= 30){  
        *DistanceCapteurD = 13 * pow(TensionCapteurD, -1);  
    }  
    else{  
        *DistanceCapteurD = -1;  
    }  
    // débogage  
    Serial.print("AD: ");  
    Serial.print(*DistanceCapteurAD); // print the distance  
    Serial.println("cm");  
    Serial.print("AG: ");  
    Serial.print(*DistanceCapteurAG); // print the distance  
    Serial.println("cm");  
    Serial.print("D: ");
```

```

Serial.print(*DistanceCapteurD); // print the distance
Serial.println("cm");
Serial.println("");
}

```

3.2.2 Capteurs de sol (si nécessaire)

Référence de conception : CDT03

Exigences client vérifiées: EXIG_LUMINOSITE

3.2.2.1 Schéma électrique des capteurs de sol

Comme mentionné en conception préliminaire, nous allons utiliser trois capteurs de contraste CNY70 avec deux capteurs devant et un capteur à l'arrière au centre (Figure 22).

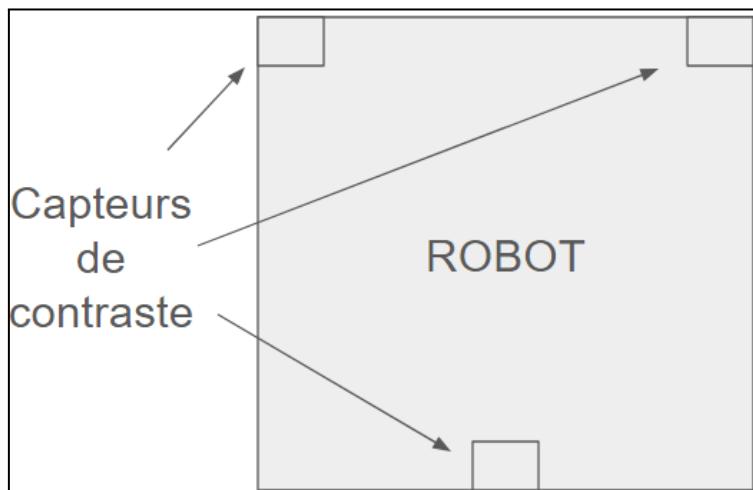


Figure 22 : Positionnement des capteurs de sol sur le robot (Vue de haut)

Pour le bon fonctionnement de ce composant (Datasheet), il faut lui ajouter deux résistances : une pour l'émetteur et l'autre pour le récepteur (Figure 23).

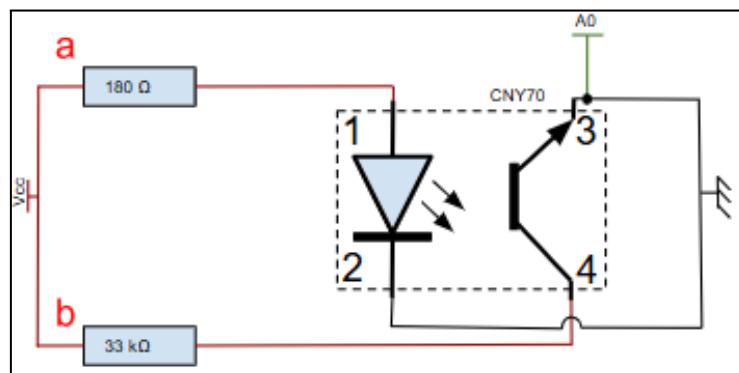


Figure 23 : Schéma électrique d'un capteur de sol (Vue interne)

Pour la résistance de l'émetteur (Figure 23.a), nous avons utilisé une simple loi d'ohm avec $V_{cc} = 5V$, la tension $V_f = 1.2V$ et le courant $I_f = 20mA$ que nous avons trouvés dans la datasheet des capteurs de contraste :

$$R_e = \frac{V_{cc} - V_f}{I_f} = \frac{5 - 1.2}{20 \cdot 10^{-3}} = 190 \Omega$$

Nous trouvons une résistance de 190Ω . Nous la normalisons à 180Ω série E96 ($\pm 1\%$).

Pour la résistance du récepteur (Figure 23.b), nous avons aussi utilisé une loi d'ohm avec $V_{cc} = 5 \text{ V}$ et le courant du collecteur = 1 mA que nous avons trouvé dans la datasheet :

$$R_r = \frac{V_{cc}}{I_c} = \frac{5}{10^{-3}} = 5000 \Omega$$

Nous trouvons une résistance de 5000Ω . Or, pour avoir un découpage net et précis entre les contrastes clairs et sombres, nous avons choisi la résistance la plus grande possible, soit $33.2 \text{ k}\Omega$ série E96 ($\pm 1\%$).

Les capteurs de sol seront reliés au shield par trois connecteurs respectifs suivis des résistances calculées au-dessus. Nous aurons donc trois montages identiques à la figure 24 pour nos trois capteurs de sol. Les broches 1 et 4 du connecteur seront reliées à la masse, la broche 2 correspond à la diode émettrice sera reliée à $V_{cc} = 5\text{V}$ par la résistance R_e et la broche 3 sera reliée à une broche analogique du microcontrôleur (ContrastX) et à V_{cc} avec la résistance R_r .

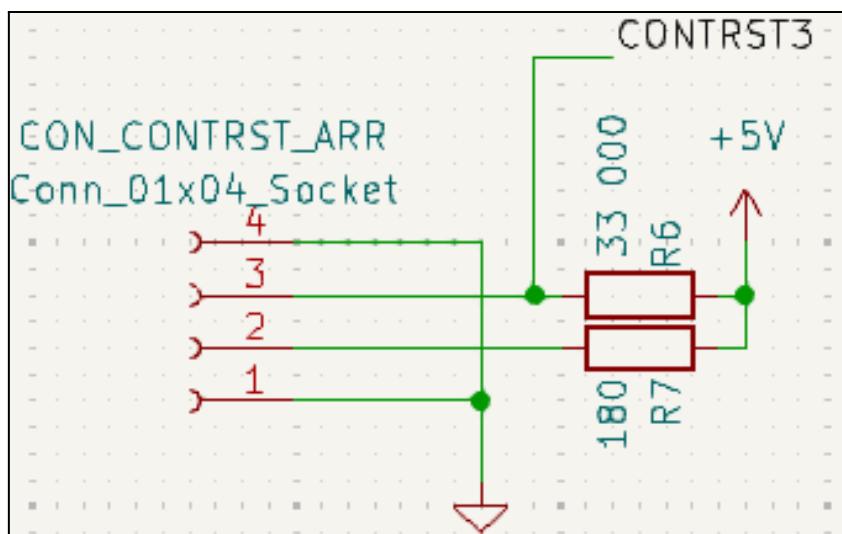


Figure 24 : Schéma électrique d'un capteur de sol sur le shield

Nous avons fait très attention au sens des connecteurs et à leurs positionnements lors du routage de cette partie.

3.2.2.2 Code informatique des capteurs de sol

La fonction AcquerirSol met à jours 3 pointeurs et émet ses données sur le port série pour le débogage.

Elle récupère le chiffre digital puis à l'aide d'une variable globale, "threshold" qui détermine si le booléen doit être vrais (si nous sommes sur une surface blanche) ou faux (si nous sommes sur une surface noir). Threshold est une variable globale pour facilité d'ajustement.

```
void AcquerirSol(bool *contrastDR,bool *contrastDG,bool *contrastA) {
    uint16_t ADC_val_far = analogRead(coll_pin_far);
```

```

uint16_t ADC_val_close = analogRead(coll_pin_close);
uint16_t ADC_val_back = analogRead(coll_pin_back);
    // threshold
*contrastDR = ADC_val_far < threshold;
*contrastDG = ADC_val_close < threshold;
*contrastA = ADC_val_back < threshold;

Serial.print("AD: ");
Serial.print(*contrastDR); // print the distance
Serial.print(" AG: ");
Serial.print(*contrastDG); // print the distance
Serial.print(" D: ");
Serial.print(*contrastA); // print the distance
Serial.println("");
}

```

3.2.3 Capteurs de télécommande

Référence de conception : CDT04

Exigences client vérifiées : EXIG_DEPLACEMENT

3.2.3.1 Schéma électrique des capteurs de télécommande

Comme mentionné dans la conception préliminaire, nous allons utiliser le récepteur IR TSOP 36438TT. Pour son bon fonctionnement et comme recommandé dans la datasheet, il faut ajouter un condensateur et une résistance (Figure 25).

Pour la résistance, nous avons appliqué une loi d'ohm avec $V_{cc} = 5V$ et un courant limite de 3 mA comme conseiller dans la datasheet :

$$R = \frac{V_{cc}}{I} = \frac{5}{3 \cdot 10^{-3}} = 1666.6 \Omega$$

Nous trouvons une résistance de 1666.6 Ω que nous normalisons à 1820 Ω série E96 ($\pm 1\%$).

Pour le condensateur, nous avons d'abord calculé la fréquence avec notre résistance et une valeur de condensateur de découplage standard 100 nF en utilisant la formule suivante :

$$f = \frac{1}{\sqrt{2\pi \cdot R \cdot C}} = \frac{1}{\sqrt{2\pi \cdot 1800 \cdot 100 \cdot 10^{-9}}} = 29.73 \text{ Hz}$$

Nous trouvons une fréquence de 29.73 Hz ce qui est très inférieur à 38 kHz correspondant à la fréquence du microcontrôleur et qui marchera parfaitement.

À cause d'un souci d'approvisionnement, nous allons remplacer le condensateur de 100 nF par un condensateur de 47 nF. Cela ne change en rien le bon fonctionnement de cet étage. La fréquence sera juste de 43.37 Hz.

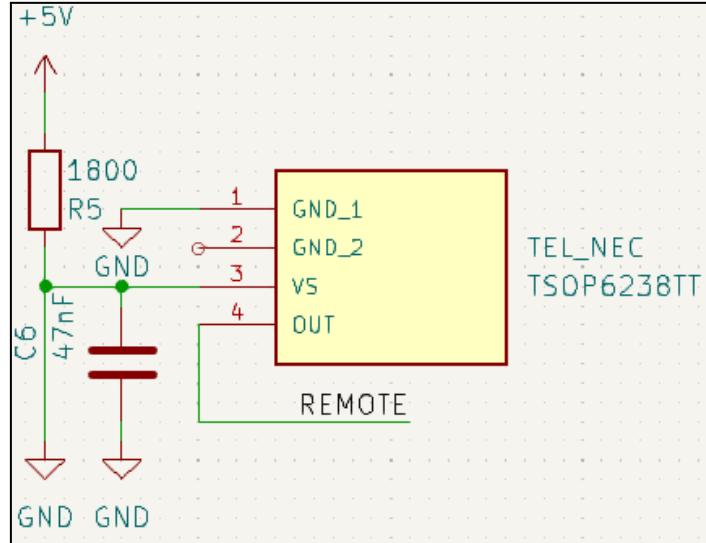


Figure 25 : Schéma électrique du capteur de télécommande

3.2.3.2 Code informatique des capteurs de télécommande

La fonction AcquerirTrameNEC utilise la bibliothèque “IrReceiver” qui s'occupe de la réception et décodage des trames reçus par le récepteur TSOP6238TT.

Quand la fonction est appelée et une trame de donnée a été reçue, la bibliothèque détermine quel protocole a été reçu et si c'est une trame NEC alors la fonction renvoie vrai et sinon faux.

```
bool AcquerirTrameNEC(){
    if (IrReceiver.decode()) {
        IrReceiver.resume(); # Réinitialiser le récepteur après avoir décodé une trame infrarouge
        protocol = IrReceiver.decodedIRData.protocol;
        return protocol == NEC;
    }
    return false;
}
```

3.3 Conception détaillée de la partie action

3.3.1 Pont en H

Référence de conception : CDT05

Exigences client vérifiées : EXIG_DEPLACEMENT

3.3.1.1 Schéma électrique du pont en H

Nous traitons du bloc action qui gère le déplacement du robot qui doit être capable de changer de direction de façon autonome en utilisant les données des capteurs et en agissant sur les actionneurs.

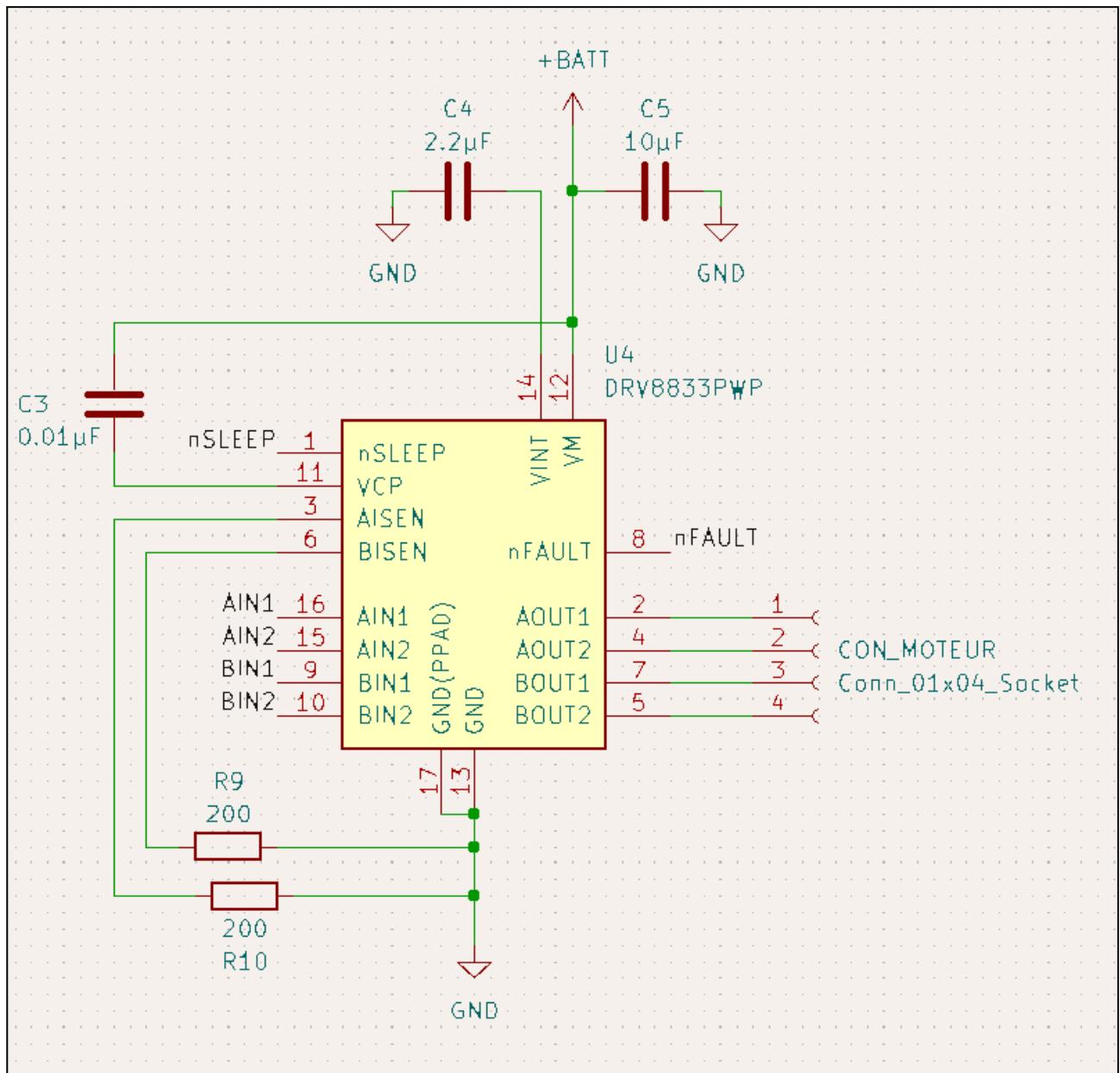


Figure 26 : Schéma électrique du pont en H

Nous utilisons le pont en H DRV8833PWP pour le contrôle des 2 moteurs. Pour cela, nous utilisons les 8 entrées nécessitant un PWM pour fonctionner : AIN1 - AIN2 - BIN1 - BIN2. Chaque PWM est généré sur une prise timer de l'ATMEGA. Nous alimentons aussi le moteur en suivant les pin VM et VCP, et le polulu en passant par le pin VINT et nous regions à la masse à l'aide des pins GND et GND(PPAD). Nous relions aussi les pins AISEN et BISEN à la masse, car leurs fonctionnements est lié à un contrôle du courant des moteurs, ce qui n'est pas nécessaire.

dans notre cas.

Aussi, nous utilisons le pin nSLEEP pour permettre une extinction des moteurs lorsque la batterie passe en dessous du seuil de coupure, et nous utilisons le pin nFAULT en cas de problème d'un moteur ou du pololu.

Pour continuer, nous avons ajouté des indicateurs lumineux, une LED verte afin de vérifier que la carte soit correctement alimentée et une LED bleu pour confirmer que la carte a reçu une trame correcte.

LED verte :

N'ayant pas d'exigence concernant les LED nous décidons de nous-même l'intensité nécessaire qu'on fixe à 100 mcd +/-20%

D'après la datasheets, pour un courant de 20 mA traversant la LED, on a :

- 1800 mcd LED-Verte

Nous devons donc calculer la valeur de la résistance.

Pour cela, nous allons utiliser la formule dans le cas où la résistance est reliée à un montage Push-Pull : $\frac{V_{cc} - (V_{cc} - V_{oh}) - V_f}{I_f}$

Or les pertes du transistor ($V_{cc} - V_{oh}$) du MCU ne sont pas prises en compte car la LED verte est directement raccordée au V_{cc} .

La valeur de I_f et V_f proviennent de la datasheet de la [LED verte](#) :

- Calcul de I_f :

On sait que pour 20 mA \rightarrow 1800 mcd

$$\text{ainsi pour 100 mcd : } I_f = \frac{20 * 100}{1800} = 1.1 \text{ mA}$$

- $V_f = 2.6 \text{ V}$

$$\frac{7.4 - 2.6}{1.1 * 10^{-3}} = 4363$$

$R (\Omega)$	R normalisée série E96 (Ω)
4 363	3 900 +/-1%

La résistance est normalisée à 3,9k Ω conformément au stock de SAE. Ainsi l'intensité final est de 108 mcd

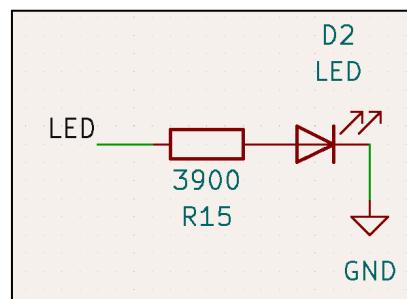


Figure 27 : Schéma électrique LED verte

LED bleu :

N'ayant pas d'exigence concernant les LED nous décidons de nous même l'intensité nécessaire. Nous choisissons donc : 100 mcd +/-20%

D'après la datasheet, 20mA fait briller la LED à 400 mCd soit, pour 100 mCd, le courant traversant la LED est égal à 5 mA.

D'après la datasheet, si le courant traversant la LED est de 5 mA, la tension aux bornes de la LED V_{AK} est d'environ 2.78 V.

Donc, d'après la formule de l'HTUT 7, la résistance en série de la LED est de :

$$R = \frac{U_r}{I_r} = \frac{V_{cc}-V_{ak}}{I_r} = \frac{5 - 2.78}{4 \times 10^{-3}} = 555 \Omega$$

R (Ω)	R normalisée série E96 (Ω)
555	560 +/-1%

3.3.1.2 Code informatique du pont en H

Pour le fonctionnement du pont en H et des moteurs, nous avons réalisé les fonctions suivantes pour contrôler le déplacement :

```
void avancer(int vitesse) {
    analogWrite(MOT_1a, vitesse);
    digitalWrite(MOT_1b, 0);
    analogWrite(MOT_2a, vitesse);
    digitalWrite(MOT_2b, 0);
} // Fait avancer le robot à 'vitesse'

void reculer(int vitesse) {
    analogWrite(MOT_1b, vitesse);
    digitalWrite(MOT_1a, 0);
    analogWrite(MOT_2b, vitesse);
    digitalWrite(MOT_2a, 0);
} // Fait reculer le robot à 'vitesse'

void pivoter(int vitesse, int dir) {
    if (dir == 0) { // TOURNE A DROITE
        analogWrite(MOT_1a, vitesse);
        digitalWrite(MOT_1b, 0);
        analogWrite(MOT_2b, vitesse);
        digitalWrite(MOT_2a, 0);
    } else { // TOURNE A GAUCHE
        analogWrite(MOT_1b, vitesse);
    }
}
```

```

        digitalWrite(MOT_1a, 0);
        analogWrite(MOT_2a, vitesse);
        digitalWrite(MOT_2b, 0);
    }
} // Le robot pivote sur lui même

void tourner(int vitesse, int dir) {
    if (dir == 1) { // TOURNE A DROITE
        analogWrite(MOT_1a, vitesse/1.4);
        digitalWrite(MOT_1b, 0);
        analogWrite(MOT_2a, vitesse);
        digitalWrite(MOT_2b, 0);
    } else { // TOURNE A GAUCHE
        analogWrite(MOT_1a, vitesse);
        digitalWrite(MOT_1b, 0);
        analogWrite(MOT_2a, vitesse/1.4);
        digitalWrite(MOT_2b, 0);
    }
} // Le robot fait un virage dans une direction (modifié 1,4 pour
l'angle de braquage)
void demi_tour(int vitesse) {
    analogWrite(MOT_1b, vitesse);
    digitalWrite(MOT_1a, 0);
    analogWrite(MOT_2b, vitesse/1.6);
    digitalWrite(MOT_2a, 0);
} // Autre forme du virage pour faciliter la prise de décision

```

Nous avons décidé de créer plusieurs petites fonctions telles que avancer(vitesse) ou reculer(vitesse) pour apporter une rapidité d'exécution (pas de switch(case) ou boucle if() à répétitions) mais aussi pour permettre une simplicité lors de la programmation de la prise de décision et des actions.

La fonction analogWrite() nous permet d'émettre un PWM sur le pin sélectionné. Aussi d'après la datasheet du polulu, le PWM gère la vitesse de rotation du moteur et le signal digital décide de l'activation du moteur (0 = moteur allumé).

La valeur maximale de la variable vitesse est 255. (valeur maximale du l'arduino)
Les valeurs 1,4 et 1,6 nous permettrons de modifier les angles de déplacement au besoin.

3.4 Conception détaillée de la partie énergie

3.4.1 Pont diviseur de tension

Référence de conception : CDT06

Exigences client vérifiées : EXIG_SECUR_BATT

Dans cette exigence, le circuit doit surveiller la tension de la batterie et couper l'alimentation de la partie puissance lorsque la tension aux bornes de celle-ci tombe en dessous de 6,7 V. Cela signifie que les moteurs, qui consomment beaucoup de courant, ne doivent plus fonctionner si la tension de la batterie passe sous ce seuil. Pour vérifier cette exigence, nous avons utilisé un amplificateur opérationnel (AOP) en mode comparateur, associé à deux ponts diviseurs.

3.4.1.1 Schéma électrique des ponts diviseurs de tension

Puisque notre amplificateur opérationnel (AOP) est alimenté par la carte Arduino, il n'est pas possible de comparer directement la tension de la batterie, car celle-ci est supérieure à la tension d'alimentation de l'AOP. Pour contourner ce problème, nous avons dimensionné un premier pont diviseur, permettant de ramener la tension de la batterie à une valeur mesurable par l'AOP.

Nous divisons la tension de la batterie par deux, passant de 7,4 V à 3,7 V, avec un courant traversant le pont diviseur $I_{\text{pont}} = 100 \mu\text{A}$

Calcul des résistances :

- $R_1 = \frac{V_{\text{batt}} - V_{\text{bat}}/2}{I_{\text{pont}}} = \frac{7.4 - 3.7}{100\mu} = 37\ 000 \text{ V}$
- $R_2 = \frac{V_{\text{bat}}/2}{I_{\text{pont}}} = \frac{3.7}{100\mu} = 37\ 000 \text{ V}$

En valeurs normalisées, nous avons choisi R_1 et R_2 à 36 kΩ.

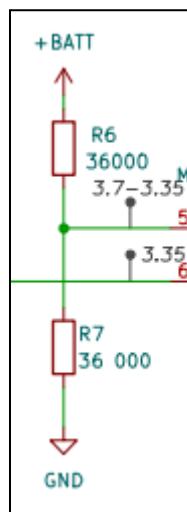


Figure 28 : Schéma électrique du premier pont diviseur de tension

Pour comparer la tension de la batterie et vérifier qu'elle ne descend pas en dessous de 6,7 V, nous avons divisé par deux la tension de la batterie. Ainsi, la tension de référence correspondant à 6,7 V sera égale à 3,35 V.

Calcul des résistances du pont diviseur :

- $R_1 = \frac{5V - V_{\text{ref}}/2}{I_{\text{pont}}} = \frac{7.4 - 3.35}{100\mu} = 16\ 500 \text{ V}$
- $R_2 = \frac{V_{\text{ref}}/2}{I_{\text{pont}}} = \frac{3.35}{100\mu} = 33\ 500 \text{ V}$

En valeurs normalisées, on choisira $R_1 = 16\text{k}\Omega$ et $R_2 = 33\text{k}\Omega$.

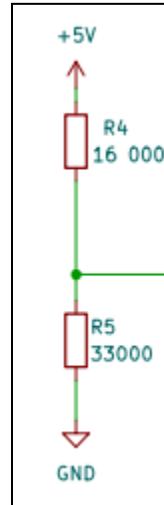


Figure 29 : Schéma électrique du second pont diviseur de tension

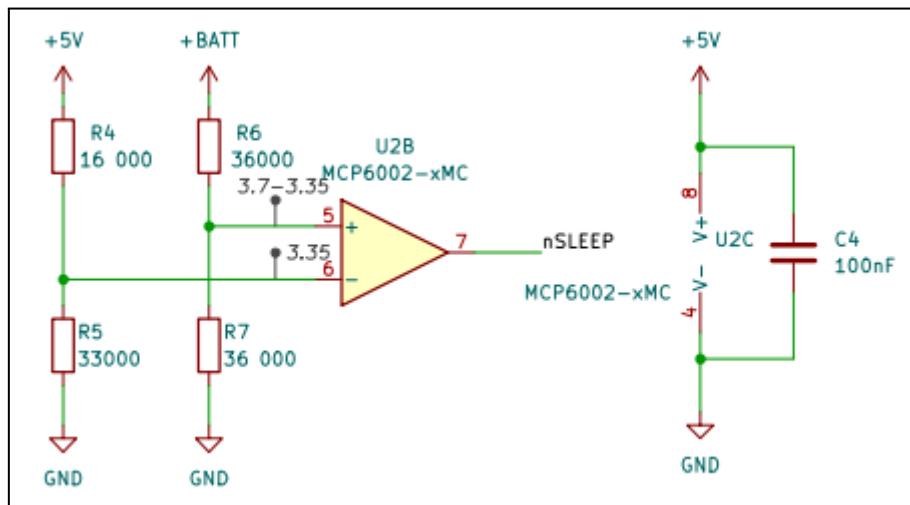


Figure 30 : Schéma électrique final

3.4.2 régulateur de tension

Référence de conception : CDT07

Exigences client vérifiées : EXIG_COMPORTEMENT, EXIG_ADVERSAIRE, EXIG_LUMINOSITE, EXIG_DEPART

3.4.2.1 Schéma électrique du régulateur de tension

Dans cette section, nous étudions le bloc fonctionnel lié à la gestion de l'énergie. Comme mentionné dans la partie préliminaire, il est nécessaire de dimensionner correctement un

régulateur de tension afin d'abaisser et de stabiliser la tension fournie par la batterie. Cet abaissement est essentiel et obligatoire pour alimenter nos capteurs ainsi que la carte Arduino. Le schéma ci-dessous montre que le régulateur est connecté à deux blocs distincts, d'où l'importance de bien le dimensionner. Voici une illustration graphique de cette explication :

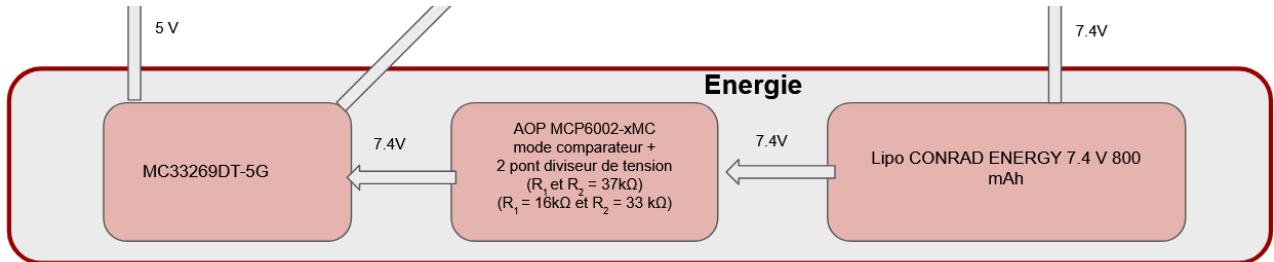


Figure 31 : Bloc fonctionnel détaillé de la partie énergie.

Voici le schéma électrique en lien avec notre bloc fonctionnel, nous avons mis en évidence notre régulateur de tension :

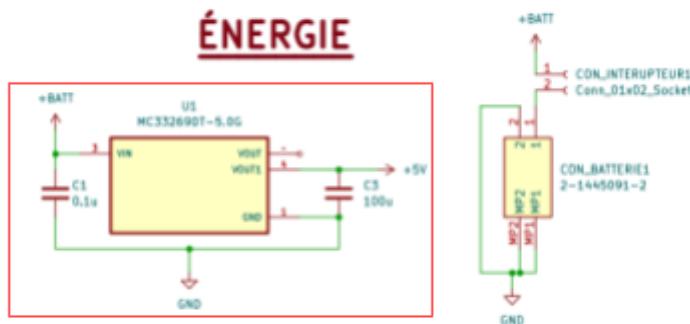


Figure 32 : schéma électrique de la partie énergie.

Les valeurs des condensateurs C_{in} (condensateur d'entrée) et C_{out} (condensateur de sortie) pour le régulateur MC33269DT-5.0 n'ont pas été calculées spécifiquement ici, car elles sont directement fournies dans la documentation technique du composant.

Le condensateur C_{in} n'est pas essentiel pour que le régulateur fonctionne correctement, mais il est conseillé de l'utiliser. Il aide à améliorer la réactivité du système et à rendre le régulateur moins sensible aux variations d'impédance de l'alimentation, surtout si la source d'entrée change beaucoup :

- La fiche technique recommande un condensateur de 0.33 µF ou plus placé aussi proche que possible de l'entrée du régulateur. Mais nous allons utiliser un $C_{in} = 0.1 \mu F$

Le C_{out} est essentiel pour la stabilité du régulateur. Selon la fiche technique le MC 33269 nécessite un condensateur de sortie externe pour garantir la stabilité du régulateur :

- La capacité recommandée pour C_{out} est au moins 10 µF.

Nous avons donc choisi un condensateur de découplage $C_{out}=100\mu F$, car c'est la valeur disponible en stock. De plus, un condensateur de plus grande capacité permet de mieux stabiliser la tension en réduisant les variations, d'où notre préférence pour un condensateur de grande valeur.

3.5 Conception détaillée de la partie traitement

3.5.1 Le microcontrôleur ATMEGA328P-PU

Référence de conception : CDT08

Exigences client vérifiées : EXIG_COMPORTEMENT

3.4.1.1 Schéma électrique du microcontrôleur

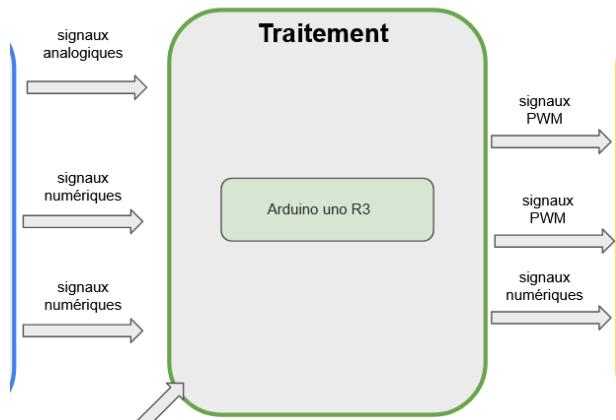


Figure 33 : bloc fonctionnel

TRAITEMENT

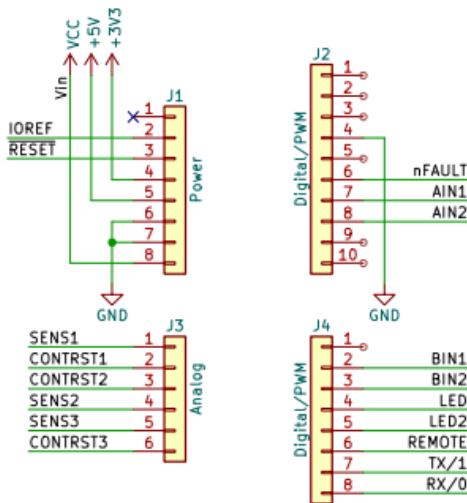


Figure 34 : schéma électrique de la partie traitement.

Pour le fonctionnement de notre robot, nous partons sur la base d'un Arduino Uno R3 (tournant sur un microprocesseur ATMEGA 328P). Nous réalisons donc le shield de cette carte

IUT Bordeaux Département GEII	Référence : RMS_DDC_EQ12 Révision : 2 – 08/10/2024	40/60
----------------------------------	---	-------

Arduino.

Ainsi, nous utilisons les broches analogiques de l'arduino pour la lecture des différents capteurs utilisés dans notre robot. Nous utilisons aussi 4 broches "timer" de l'arduino pour le contrôle du polulu (qui nécessite des signaux PWM).

3.3.1.2 Code informatique du microcontrôleur

Dans cette partie, nous traitons les différentes fonctions utilisées pour programmer notre robot. Nous commençons par les fonctions acquisitions avant de continuer sur celle d'action pour finir sur le void loop() qui nous sert de fonction principale.

```

1 #include <arduino.h>
2 #include "IRremote.hpp"
3 #define IR_RECEIVE_PIN 2

```

Nous commençons par inclure les librairies utilisées dans notre code ainsi que la définition du pin du récepteur IR.

L'import de la librairie arduino.h est rappelé car le code a été compilé en .cpp et non en .ino.

```

//Gestion des I/Os
constexpr int LED_BLEUE = 4;
constexpr int MOT_1a = 5;
constexpr int MOT_1b = 6;
constexpr int MOT_2a = 10;
constexpr int MOT_2b = 11;
constexpr int MOTFAULT = 12;
constexpr int coll_pin_far = 14;
constexpr int coll_pin_close = 15;
constexpr int coll_pin_back = 16;
constexpr int capteurAD = 17;
constexpr int capteurAG = 18;
constexpr int capteurD = 19;

```

Nous continuons par la définition de nos broches utilisées. Nous créons des variables constexpr, natives du cpp, qui ont la particularité d'être compilées au début du code et non lors de leur appel, cela permet d'éviter un éventuel délai supplémentaire.

```
//Gestion des constantes
float DistanceCapteurAD;
float DistanceCapteurAG;
float DistanceCapteurD;
unsigned int threshold_contrast = 30;
int threshold_distance = 30;
uint16_t protocol = IrReceiver.decodedIRData.protocol;
bool RobotActif = false;
```

Nous initialisons ensuite nos différentes variables, utilisées à travers le code.

```
bool AcquerirTrameNEC(){
    if (IrReceiver.decode()) {
        IrReceiver.resume(); // Re init le recept apres decodage trame nec
        protocol = IrReceiver.decodedIRData.protocol;
        return protocol == NEC;
    }
    return false;
}
```

Nous créons la fonction AcquerirTrameNEC() pour la lecture du capteur IR ainsi que le décodage d'une éventuelle trame.

Robot Mini-Sumo (RMS)

```
void acquerirAdversaires(float *DistanceCapteurAD, float *DistanceCapteurAG, float *DistanceCapteurD) {
    // NE PAS APPELER PLUS DE 50 FOIS / sec
    const float TensionCapteurAD = analogRead(pin:capteurAD) * 0.0048828125;
    const float TensionCapteurAG = analogRead(pin:capteurAG) * 0.0048828125;
    const float TensionCapteurD = analogRead(pin:capteurD) * 0.0048828125;
    if (13 * pow(x:TensionCapteurAD, y:-1) <= 30){
        *DistanceCapteurAD = 13 * pow(x:TensionCapteurAD, y:-1);
    }
    else{
        *DistanceCapteurAD = -1;
    }

    if (13 * pow(x:TensionCapteurAG, y:-1) <= 30){
        *DistanceCapteurAG = 13 * pow(x:TensionCapteurAG, y:-1);
    }
    else{
        *DistanceCapteurAG = -1;
    }

    if ( 13 * pow(x:TensionCapteurD, y:-1) <= 30){
        *DistanceCapteurD = 13 * pow(x:TensionCapteurD, y:-1);
    }
    else{
        *DistanceCapteurD = -1;
    }
}
```

La fonction acquerirAdversaires(pointers) nous permet de mesurer la distance mesuré par chaque capteurs toutes les 20ms en les retournant à travers des pointers.

```
void acquerirSol(bool *contrastDR, bool *contrastDG, bool *contrastA) {
    uint16_t ADC_val_far = analogRead(coll_pin_far);
    uint16_t ADC_val_close = analogRead(coll_pin_close);
    uint16_t ADC_val_back = analogRead(coll_pin_back);

    *contrastDR = ADC_val_far < threshold_contrast;
    *contrastDG = ADC_val_close < threshold_contrast;
    *contrastA = ADC_val_back < threshold_contrast;
}
```

La fonction acquerirSol(pointers) nous permet de détecter en cas de passage d'un capteur sur la bande blanche en extérieur du dohyo. Nous retournons les valeurs à travers des pointers.

```
void avancer(int vitesse) {
    analogWrite(pin: MOT_1a, vitesse);
    digitalWrite(pin: MOT_1b, val: 0);
    analogWrite(pin: MOT_2a, vitesse);
    digitalWrite(pin: MOT_2b, val: 0);
}

void reculer(int vitesse) {
    analogWrite(pin: MOT_1b, vitesse);
    digitalWrite(pin: MOT_1a, val: 0);
    analogWrite(pin: MOT_2b, vitesse);
    digitalWrite(pin: MOT_2a, val: 0);
}
```

Ensuite, nous avons les fonctions pour avancer et reculer, chaque fonction envoie un PWM sur une entrée moteur du pont en H, en échangeant la broche de chaque moteur, nous changeons le sens de rotation.

```
void pivoter(int vitesse, int dir) {
    if (dir == 0) { // TOURNE A DROITE
        analogWrite(pin: MOT_1a, vitesse);
        digitalWrite(pin: MOT_1b, val: 0);
        analogWrite(pin: MOT_2b, vitesse);
        digitalWrite(pin: MOT_2a, val: 0);
    } else { // TOURNE A GAUCHE
        analogWrite(pin: MOT_1b, vitesse);
        digitalWrite(pin: MOT_1a, val: 0);
        analogWrite(pin: MOT_2a, vitesse);
        digitalWrite(pin: MOT_2b, val: 0);
    }
}

void tourner(int vitesse, int dir) {
    if (dir == 1) { // TOURNE A DROITE
        analogWrite(pin: MOT_1a, val: vitesse/1.4);
        digitalWrite(pin: MOT_1b, val: 0);
        analogWrite(pin: MOT_2a, vitesse);
        digitalWrite(pin: MOT_2b, val: 0);
    } else { // TOURNE A GAUCHE
        analogWrite(pin: MOT_1a, vitesse);
        digitalWrite(pin: MOT_1b, val: 0);
        analogWrite(pin: MOT_2a, val: vitesse/1.4);
        digitalWrite(pin: MOT_2b, val: 0);
    }
}
```

Les fonctions pivoter et tourner permettent au robot de faire un virage dans une direction ou une autre, ou alors de pivoter sur lui-même.

Robot Mini-Sumo (RMS)

```
void setup(){
    pinMode(capteurAG, mode: INPUT);
    pinMode(capteurAD, mode: INPUT);
    pinMode(capteurD, mode: INPUT);
    pinMode(coll_pin_far, mode: INPUT);
    pinMode(coll_pin_close, mode: INPUT);
    pinMode(coll_pin_back, mode: INPUT);
    pinMode(MOT_1a, mode: OUTPUT);
    pinMode(MOT_1b, mode: OUTPUT);
    pinMode(MOT_2a, mode: OUTPUT);
    pinMode(MOT_2b, mode: OUTPUT);
    pinMode(LED_BLEUE, mode: OUTPUT);
    IrReceiver.begin( aReceivePin: IR_RECEIVE_PIN, aEnableLEDFeedback: ENABLE_LED_FEEDBACK);
}
```

La fonction void setup() nous sert à initialiser les I/Os de l'arduino.

```
void loop(){ // On s'en sert en gestionnaire des interruptions (capteur sol et millis)
if (AcquerirTrameNEC() || RobotActif == true){
    if (firstMove == false) {
        tourner(vitesse: 255, dir: 1); // Virage à gauche maximum
        delay(ms: 500); // RECALCULER LES VALEURS DE VIRAGES ET DE DELAY
    }
    float DistanceCapteurAD, DistanceCapteurAG, DistanceCapteurD;
    RobotActif = true;
    bool contrastDR, contrastDG, contrastA;
    acquerirSol(&contrastDR, &contrastDG, &contrastA);
    digitalWrite(pin: LED_BLEUE, val: 1);
    if (contrastA) {
    } else if (contrastDG) {
        tourner(vitesse: 255, dir: 0);
    } else if (contrastDR) {
        tourner(vitesse: 255, dir: 1);
    } else {
        if (millis() % 20 == 0){ // Passage en mode prise de décision des actions (jusqu'à une interruption)
            acquerirAdversaires(&DistanceCapteurAD, &DistanceCapteurAG, &DistanceCapteurD);
            if (DistanceCapteurAD <= threshold_distance && DistanceCapteurAG <= threshold_contrast){
                avancer(vitesse: 255);
            } else if (DistanceCapteurAD <= threshold_distance) {
                tourner(vitesse: 255, dir: 0);
            } else if (DistanceCapteurAG <= threshold_distance) {
                tourner(vitesse: 255, dir: 1);
            } else if (DistanceCapteurD <= threshold_distance) {
                pivoter(vitesse: 255, dir: 0);
            } else {
                pivoter(vitesse: 255, dir: 1);
            }
        }
    }
}
```

Notre fonction void loop() sert au comportement général du robot. Nous commençons d'abord par

Robot Mini-Sumo (RMS)

une attente de réceptions de la trame NEC, (environ 130000 vérifications par seconde) avant de bloquer la fréquence du code à 50Hz avec une boucle if(`millis() % 20 == 0`) lorsque la trame est reçue.

Aussi, nous partons forcément avec notre adversaire à notre gauche, le code prévoit donc de faire un virage vers le robot adverse avant de rentrer dans sa boucle de recherche-attaque principale. Cette boucle consiste en un check des capteurs de positions adverses avant d'avancer sur l'adversaire si celui-ci est repéré.

De plus, nous utilisons les capteurs sol en interruptions (avant la boucle if (`millis() % 20 ==0`)) ce qui permet d'interrompre le mouvement en cours si un capteur sol se déclenche.

Ce code sera à modifier au besoin, si les seuils de détection ne sont pas bons ou si les rapport dans les virages ne fonctionnent pas.

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 Dérisquage des capteurs de contraste

Référence de la simulation : SIM_001

Exigences client vérifiées : EXIG_LUMINOSITE

But de l'essai : Nous désirons nous assurer que nous savons utiliser les capteurs de contraste.

Fichiers : Aucun

Programme : Test_capteur_contraste_V5

Procédure de simulation :

Afin de tester les capteurs de contraste nous avons :

1. Câbler le capteurs comme montré sur la figure 34
2. Télécharger le code de test "Test_capteur_contraste_V5"
3. Connecter la carte Arduino à une ordinateur avec un câble USB
4. Ouvrir l'éditeur Arduino, sélectionner le port COM sur lequel la carte est connecté
5. Appuyer sur le bouton flasher
6. Ouvrir le moniteur série et s'assurer que la vitesse de communication est de 9600 baud/s

Après avoir câblé le montage suivant :

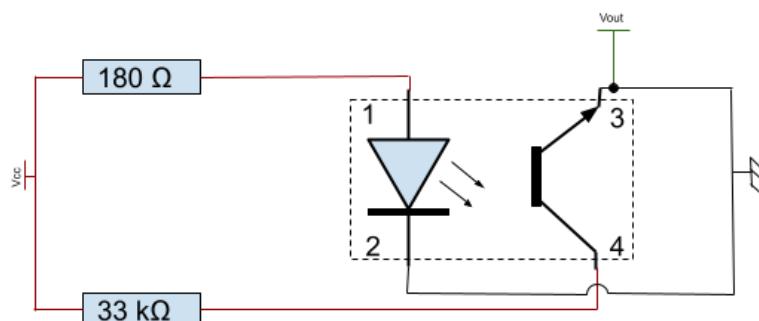


Figure 35 : Schéma du montage d'un capteur de contraste

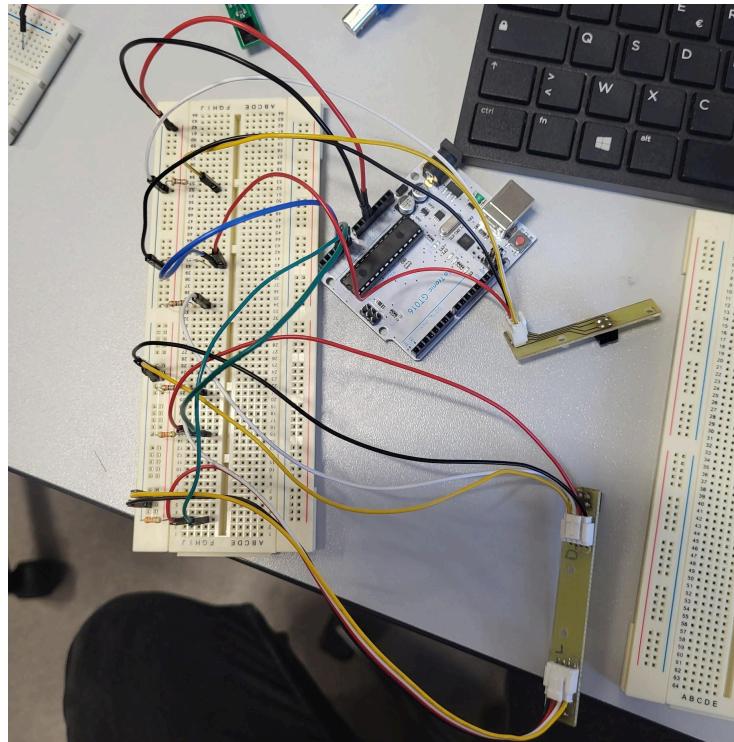


Figure 36 : Montage sur breadboard des capteurs de contraste

Résultats attendus :

Nous voulons recevoir un 1 logique quand nous sommes au-dessus d'une surface réfléchissante et un 0 logique sinon. Les capteurs doivent aussi renvoyer une valeur correcte indépendamment de ce que renvoient les autres capteurs.

Résultats obtenus :

Nous avons bien récupéré un 1 logique quand notre surface en face d'un capteur était blanche et un 0 logique quand le capteur était noir.

Grandeur	Valeur mesurée	Conf/Non conf.
Valeur logique pour une surface blanche	1	Conforme
Valeur logique pour une surface noire	0	Conforme

Statut de l'essai : l'essai a confirmé que notre solution technique est conforme aux exigences client.

Problèmes rencontrés :

Après le test initial nous avons remarqué que les trois capteurs renvoient tous la même valeur et que deux des capteurs n'avaient aucun effet sur cette valeur. Après une analyse visuelle et un test de continuité, nous nous sommes rendus compte que les lignes d'alimentation et ground n'étaient pas continue sur toute la longueur de breadboard. Les deux broches sur lesquelles étaient censées être branchées les capteurs qui n'affectent pas les données reçues étaient donc flottantes. Après avoir remplacé le breadboard avec un breadboard qui possède des lignes d'alimentation et ground continue, le montage a fonctionné comme prévu.

4.2 Dérisquage des capteurs adversaires

Référence de la simulation : SIM_002

Exigences client vérifiées : EXIG_ADVERSAIRE

But de l'essai : Nous désirons nous assurer que nous savons utiliser les capteurs de distance que nous voulons utiliser pour détecter nos adversaires.

Fichiers : Test_capteur_distance_V3_finnal_

Procédure de simulation :

Afin de tester les capteurs nous avons:

1. Câbler les trois capteurs en connectant Vcc à la broche 5V de l'arduino, GND à la broche GND de l'arduino et les cable de donnees sur les broche A0, A1et A2
2. Télécharger le code de test "Test_capteur_contraste_V5"
3. Connecter la carte Arduino à une ordinateur avec un câble USB
4. Ouvrir le fichier "Test_capteur_contraste_V5" l'éditeur Arduino et sélectionner le port COM sur lequel la carte est connecté
5. Appuyer sur le bouton flasher
6. Ouvrir le moniteur série et s'assurer que la vitesse de communication est de 9600 baud/s

Vidéo du test des capteurs adversaires: Prototypage_Cap_Adversaire.mp4

Résultats attendus :

Nous nous attendons à recevoir une distance comprise entre 2 et 30 cm et une valeur de -1 si la distance est supérieure à 30 cm

Grandeur	Valeur attendue
Valeur en centimètre à 6 cm	2 cm
Valeur en centimètre à 30 cm	30 cm
Valeur en centimètre à 31 cm	-1 cm

Résultats obtenus :

Nous avons bien récupéré les valeurs correspondant aux distances réelles et la valeur -1 pour les valeurs supérieures à ce que notre capteur est capable de détecter.

Grandeur	Valeur mesure	Conf/Non conf.
valeur en centimètre à 6 cm	2 cm	Conforme
valeur en centimètre à 30 cm	30 cm	Conforme
valeur en centimètre à 31 cm	-1 cm	Conforme

Statut de l'essai :

L'essai a confirmé que notre solution technique est conforme aux exigences client.

Problèmes rencontrés :

Aucun problème n'a été rencontré au cours de ce test.

4.3 Dérisquage des capteurs analogiques

Référence de la simulation : SIM_003

Exigences client vérifiées : EXIG_ADVERSAIRE, EXIG_LUMINOSITE

But de l'essai : Nous voulons nous assurer que tous nos capteurs sont capables de fonctionner ensemble.

Fichiers : Fonctions_ACK

Procédure de simulation : Test capteur analogiques

Afin de tester les capteurs nous avons:

7. Câbler les capteurs comme décrit dans le schéma ci-dessous.
8. Télécharger le code de test "Fonctions_ACK.ino"
9. Connecter la carte Arduino à une ordinateur avec un câble USB
10. Ouvrir le fichier "Fonctions_ACK.ino" dans l'éditeur Arduino et sélectionner le port COM sur lequel la carte est connecté
11. Appuyer sur le bouton flasher
12. Ouvrir le moniteur série et s'assurer que la vitesse de communication est de 9600 baud/s

vidéo du test des capteurs analogiques:

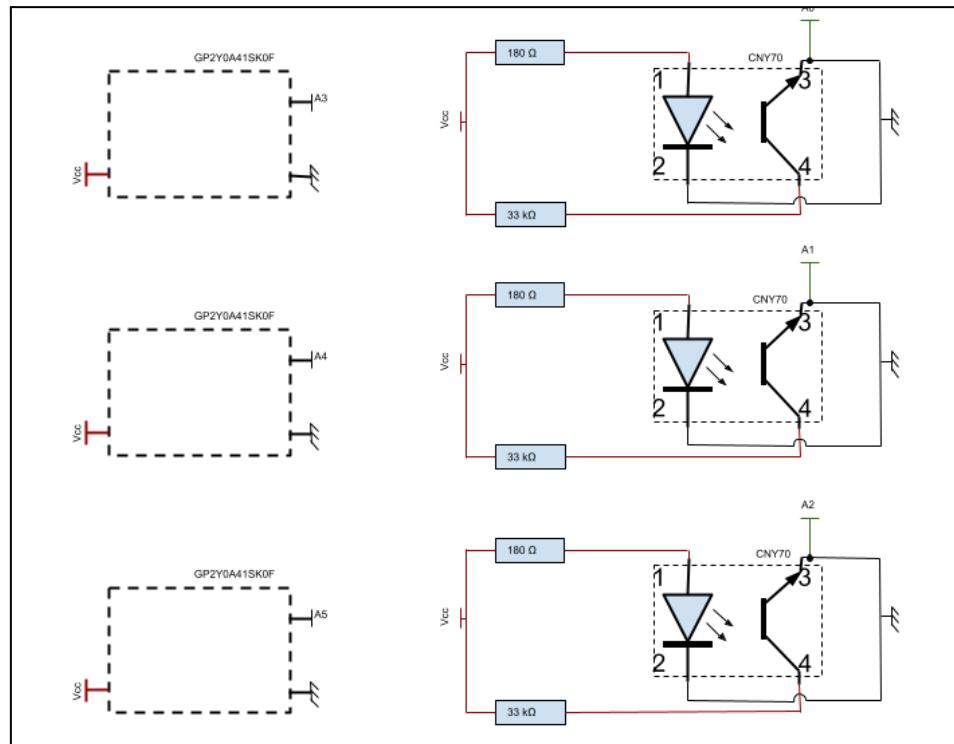


Figure 37 : Schéma du test de l'étage acquisition

Résultats attendus :

nous nous attendons à voir les valeurs que nous avions aux deux tests précédents.

Résultats obtenus :

Nous avons bien retrouvé les valeurs que nous avons trouvées dans les tests indépendants.

Statut de l'essai :

Faire fonctionner tous nos capteurs ensemble n'a pas altéré le fonctionnement général de la partie Acquisition. Nous sommes donc conformes aux exigences EXIG_ADVERSAIRE et EXIG_LUMINOSITE imposées par le client.

Problèmes rencontrés :

Nous n'avons pas eu de problème pendant les tests de l'étage acquisition.

4.3 Dérisquage du récepteur démodulateur

Référence de la simulation : SIM_004

Exigences client vérifiées : EXIG_COMPORTEMENT, EXIG_DEPART

But de l'essai : Nous voulons comprendre comment utiliser la librairie que nous allons installer sur le robot.

Fichiers :  Test_recepteur_IR

Procédure de simulation :

Afin de nous familiariser avec la librairie nous avons:

1. Câbler les capteurs comme décrit dans le schéma ci-dessous.
2. Télécharger le code de test "récepteur_infrarouge.ino"
3. Connecter la carte Arduino à un ordinateur avec un câble USB
4. Ouvrir le fichier "récepteur_infrarouge.ino" dans l'éditeur Arduino et sélectionner le port COM sur lequel la carte est connectée
5. Appuyer sur le bouton flasher
6. Ouvrir le moniteur série et s'assurer que la vitesse de communication est de 9600 baud/s

Robot Mini-Sumo (RMS)

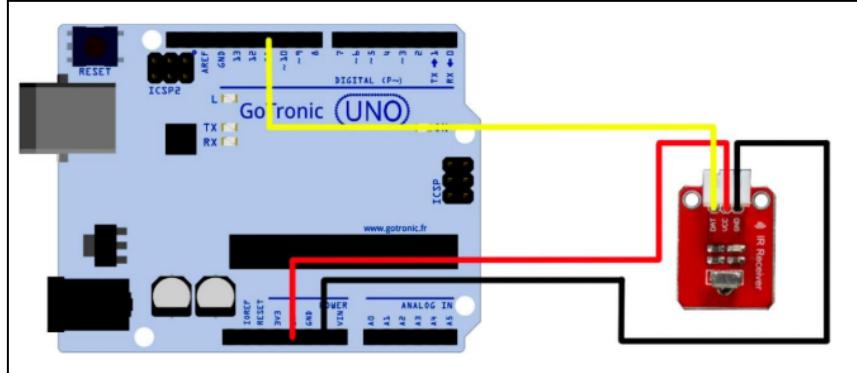


Figure 38: Schéma du test de réception IR

Résultats attendus :

Nous renverrons un booléen, vrai si la trame est bien une trame NEC et faux sinon.

Grandeur	Valeur attendue
Récupération du protocole	"NEC"
booléen "démarrage"	1

Résultats obtenus :

Nous avons bien retrouvé les résultats désirés.

Grandeur	Valeur mesurée	Conf/Non conf.
Récupération du protocole	"NEC"	Conforme
booléen "démarrage"	1	Conforme

Statut de l'essai :

Nous avons bien décodé la trame et déterminé que le robot devait démarrer. Nous sommes donc conformes à l'exigence.

Problèmes rencontrés :

Nous n'avons pas rencontré de problème notable pendant ce test.

4.4 Dérisquage du pont en H

Référence de la simulation : SIM_004

Exigences client vérifiées : EXIG_DEPLACEMENT

But de l'essai : Nous désirons nous assurer que nous savons utiliser le pont en H pour déplacer le robot SUMO.

Fichiers :

Procédure de simulation :

IUT Bordeaux Département GEII	Référence : RMS_DDC_EQ12 Révision : 2 – 08/10/2024	53/60
----------------------------------	---	-------

Robot Mini-Sumo (RMS)

Afin de tester le pont en H nous avons :

1. Câbler la carte arduino et la carte du pont en H au moteur comme ci-dessous:

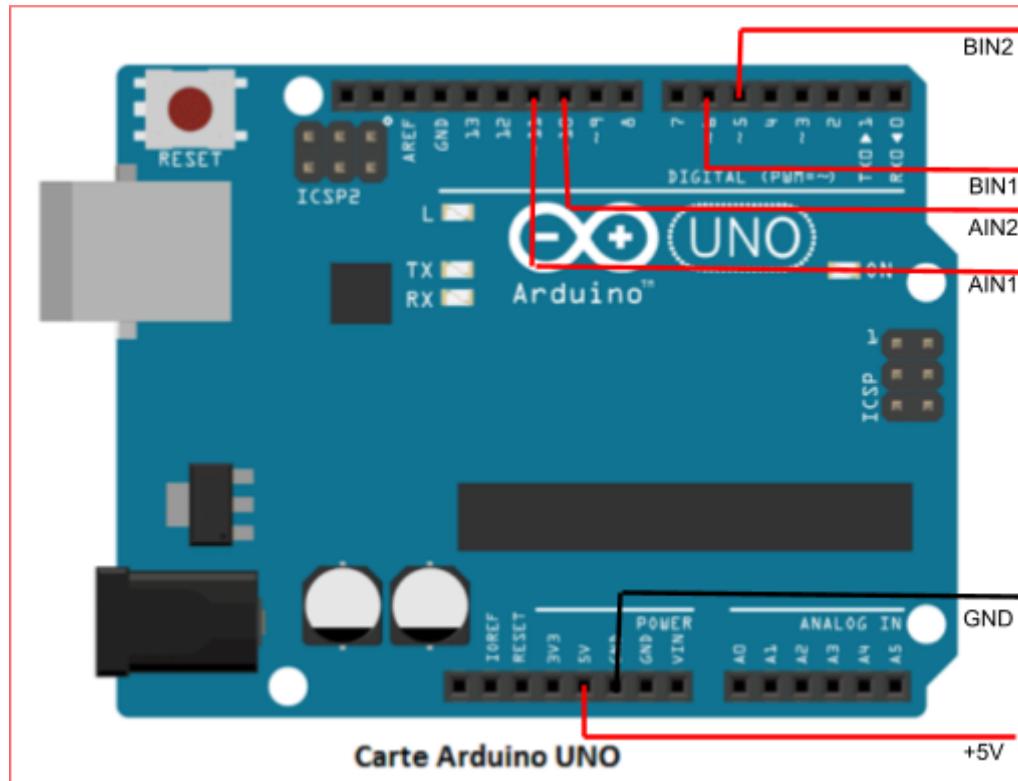


Figure 39 : Câblage de la carte arduino

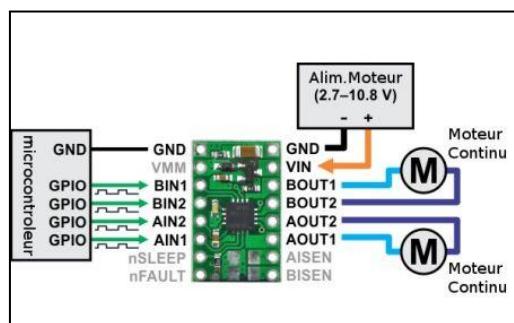


Figure 40 : Câblage du pont en H

2. Téléverser le code de test
3. Sélectionner le port COM sur lequel la carte est connecté
4. Appuyer sur le bouton flasher
5. Observer le bon fonctionnement des roues

Résultats attendus :

Nous nous attendons ce que les roues puissent tourner à la même vitesse pour avancer, en sens opposé pour faire demi-tour, en sens inverse pour reculer et à des vitesses différentes pour tourner

Résultats obtenus :

Nous avons bien observé et confirmé les bons déplacements du robot SUMO

Vidéo statuant la conformité de l'exigence : [dériskage du pont en H](#)

Statut de l'essai : l'essai a confirmé que notre solution technique est conforme aux exigences client.

Problèmes rencontrés :

Aucun problème n'a été rencontré au cours de ce test.

4.5 Dériskage de la batterie

Référence de la simulation : SIM_05

Exigences client vérifiées : EXIG_AUTONOMIE et EXIG_SECUR_BATT

But de l'essai :

Trouver une solution et montrer qu'elle permet de résoudre le problème du risque de brancher la batterie à l'envers, ce qui pourrait provoquer un court-circuit. Pour cela, nous avons choisi d'utiliser un connecteur de batterie adapté.

Procédure de simulation :

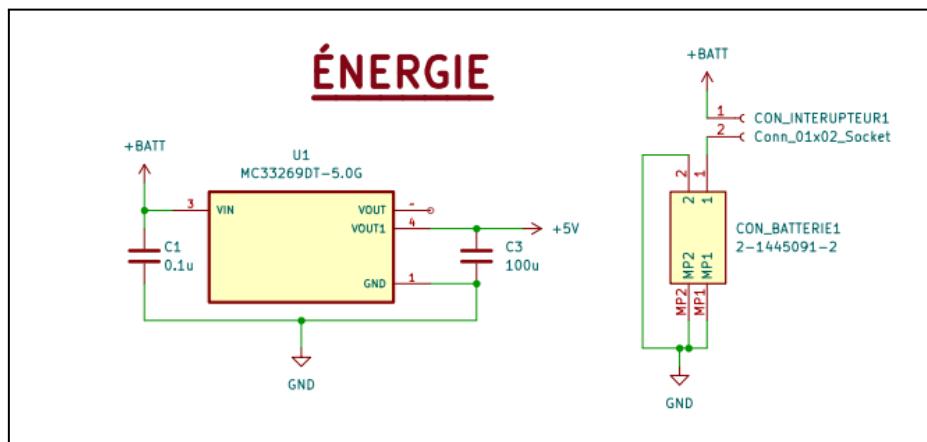
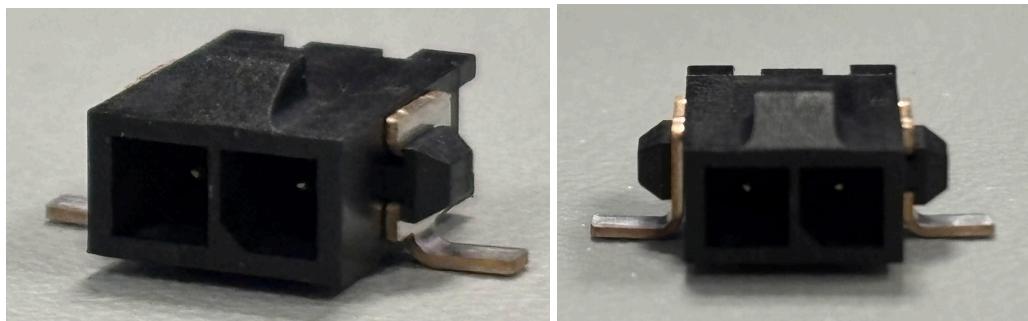


Figure 41 : Schéma électrique de la batterie et de son connecteur

Résultats attendus :

Grandeur	Valeur attendue	Tolérance
Connecteur batterie	la batterie est bien connecté	non compris

Résultats obtenus :**Figure 42 : visualisation du capteur**

Nous remarquons que ce connecteur dispose d'un détrompeur, ce qui permet d'avoir zéro erreur de branchage à l'envers.

Grandeur	Valeur mesurée	Conf/Non conf.
AMP-TE CONNECTIVITY 2-1445091-2	la batterie est bien connecté	Conforme

Statut de l'essai : conforme

Problèmes rencontrés :

Aucun problème.

4.6 Dérisquage du pont diviseur de tension et de l'AOP

Référence de la simulation : SIM_006

Exigences client vérifiées : EXIG_SECUR_BATT

But de l'essai : L'objectif de cet essai est de vérifier que le montage fonctionne correctement en contrôlant le niveau de la batterie :

- Lorsque la tension de la batterie dépasse +6,7 V, le montage doit envoyer +5 V à l'entrée **nSLEEP** du pont en H, permettant ainsi aux moteurs de fonctionner.
- Lorsque la tension de la batterie est inférieure à cette valeur, le montage doit envoyer +0 V à l'entrée **nSLEEP**, ce qui coupe l'alimentation des moteurs.

Procédure de simulation :

Nous avons refait sur une breadboard le montage suivant :

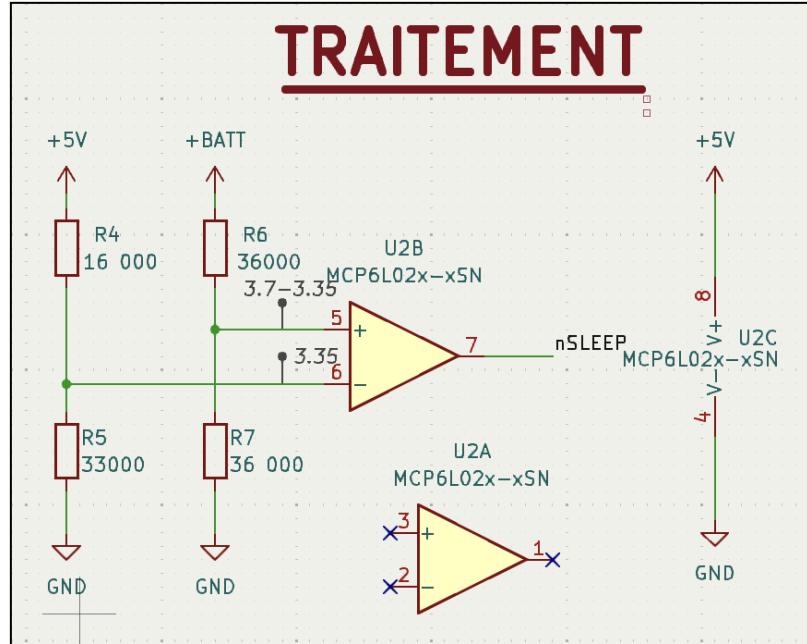


Figure 43 : Montage sur KiCad

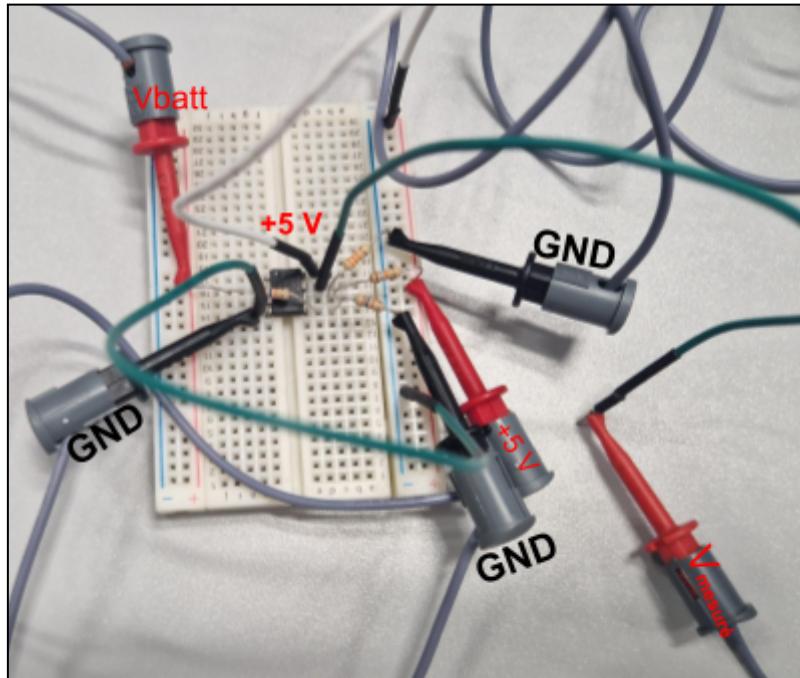


Figure 44 : Montage sur breadboard

Que nous avons alimenter avec une génératrice de tension continue avec deux sources :

- +5 V pour simuler la tension du régulateur de tension est donc celle qui alimente
- V_{batt} pour simuler la tension de la batterie

Robot Mini-Sumo (RMS)

Avec un multimètre que nous allons utiliser en mode voltmètre on va mesurer la tension sortant de l'AOP en faisant varier celle V_{batt} grâce à cela nous pouvons vérifier le bon respect de l'exigence.

Résultats attendus :

V_{batt}	$V_{mesuré}$
supérieur à +6.7 V	+5 V
inférieur à +6.7 V	0 V

Résultats obtenus :

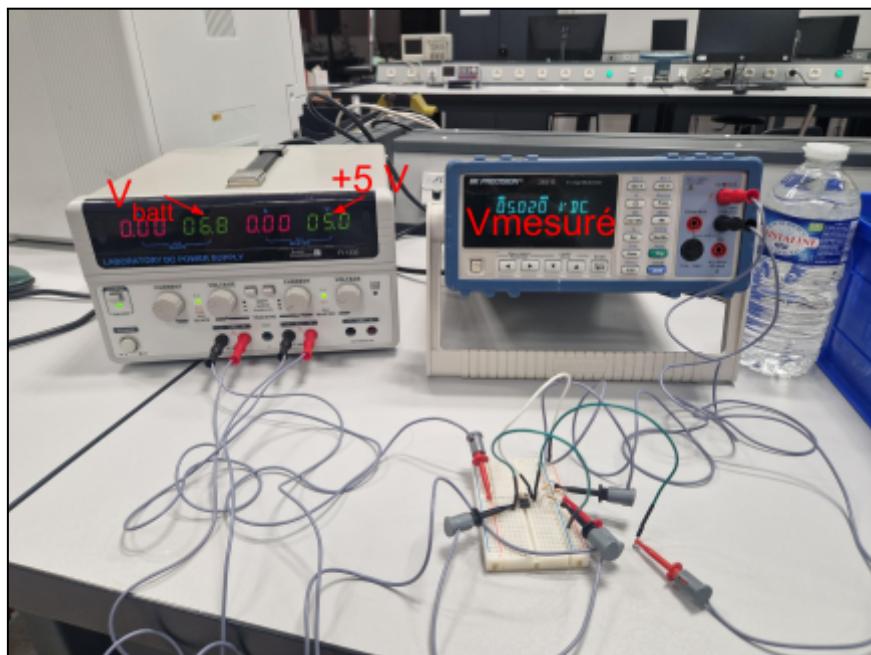


Figure 45 : essai avec $V_{batt} > +6,7 \text{ V}$

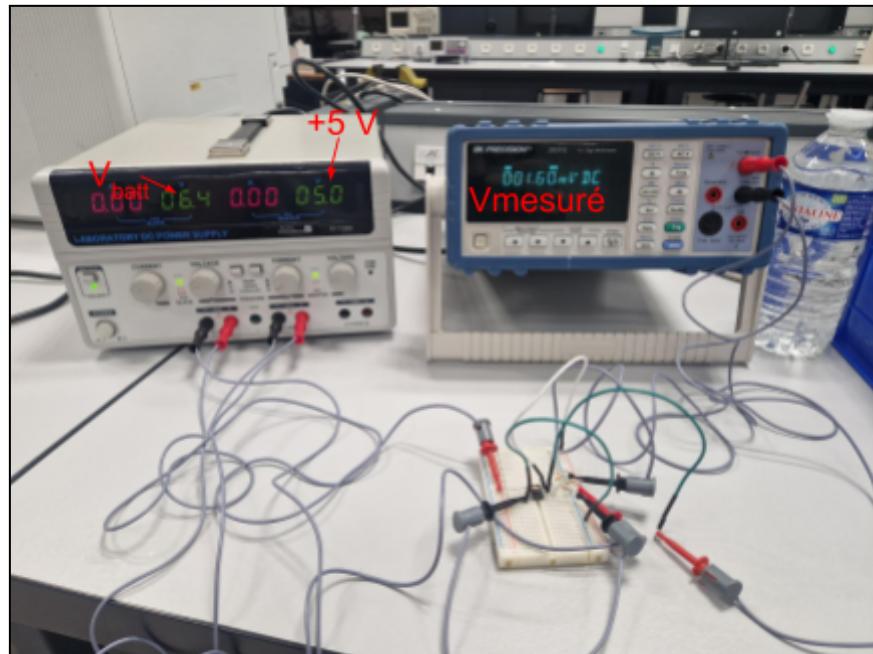


Figure 46 : essai avec $V_{batt} < +6,7 \text{ V}$

V_{batt}	$V_{mesuré}$
supérieur à +6.7 V	+5 V
inférieur à +6.7 V	0 V

Statut de l'essai : Conforme.

Problèmes rencontrés : Aucun problème.

Conclusion de la simulation / prototypage rapide du produit

Les différents dérisques et prototypages rapides ont permis de lever des doutes sur les éventuels problèmes. Nous n'avons rencontré aucunes non conformités et validons les choix et dimensionnement de la phase de conception.

5. Conclusion de la conception du produit

En conclusion, la phase de conception du produit a permis de satisfaire l'ensemble des exigences définies dans le cahier des charges. Aucune non-conformité n'ayant été

relevée lors de l'étape de conception détaillée, nous sommes en mesure de passer à la phase de fabrication en toute confiance. 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_AUTONOMIE	Conception Conception/Fab.	CPR_01,CPR_08,CDT01,SIM_05	Conf.
EXIG_SECUR_BATT	Conception Dérisque Conception/Fab.	CPR_01,CPR_09,CDT06,SIM_006	Conf.
EXIG_ADVERSAIRE	Conception Dérisque Conception/Fab.	CPR_01,CPR_02,CPR_05,CPR_07 CPR_09,CPR_10,CDT01,CDT02,C DT07,SIM_002,SIM_003,SIM_05	Conf.
EXIG_LUMINOSITE	Conception Dérisque Conception/Fab.	CPR_01,CPR_07,CPR_09,CPR_10 ,CDT01,CDT03,CDT07,SIM_001,SI M_003,SIM_004	Conf.
EXIG_DEPART	Conception Dérisque Conception/Fab.	CPR_01,CPR_07,CPR_09,CPR_10 ,CDT01,CDT07,SIM_004	Conf.
EXIG_DEPLACEMENT	Conception Dérisque Débogage	CPR_01,CPR_04,CPR_06,CPR_09 ,CPR_11,CPR_12,CPR_13,CDT01, CDT05,SIM_004,	Conf.
EXIG_COMPORTEMENT	Conception Dérisque Conception/Fab.	CPR_01,CPR_07,CPR_09,CPR_13 ,CDT01,CDT07,CDT08,SIM_004	Conf.