INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech | LEnsE Laboratoire d'Enseignement Expérimental

Institut d'Optique Graduate School
Interfaçage Numérique

INTERFAÇAGE NUMÉRIQUE

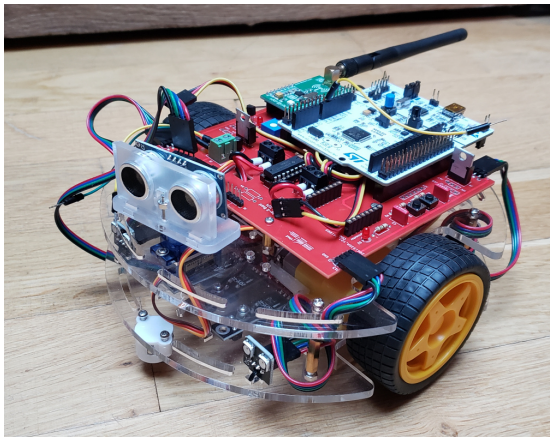
Travaux Pratiques

Semestre 6

Robotique et systèmes embarqués

---

4 séances



Ce sujet est disponible au format électronique sur le site du LEnsE - https ://lense.institutoptique.fr/ dans la rubrique Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Robotique.

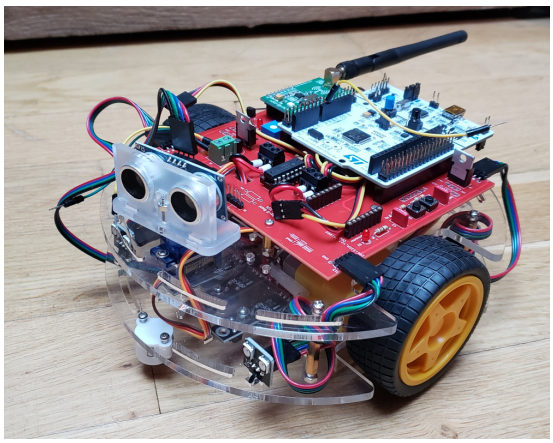
Robotique et systèmes embarqués

À l'issue des séances de TP concernant le **bloc de robotique**, les étudiant·es seront capables de :

— Développer et mettre en œuvre une **solution d'électronique embarquée** pour **mettre en mouvement un robot**
— Concevoir un **programme embarqué** permettant de **rendre autonome les mouvements d'un robot**

Objectifs du mini-projet

L'objectif principal de ce mini-projet est de **développer le code embarqué d'une plateforme robotique** lui permettant de se déplacer de manière autonome le long d'une ligne sans percuter d'obstacle.



Vous aurez à votre disposition une **maquette** basée sur un robot Joy-It Car. Cette maquette est pilotée par une carte Nucléo (contenant un microcontroleur).

Déroulement du bloc

La liste des étapes à suivre pour la réalisation du programme embarqué de la plateforme robotique est donnée à titre indicatif. L'ordre et le choix des différentes étapes sont laissés à l'appréciation des différents binômes.

Afin de faciliter la réutilisation des codes, il pourra être intéressant de définir des fonctions pour le pilotage des différents éléments.

Séance 1 / MBED et Nucléo-STM32 (sans maquette ! !)

Le sujet de cette séance est fourni dans un document annexe, disponible aussi sur le site du LEnsE - https ://-lense.institutoptique.fr/ dans la rubrique Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Intro MBED et STM32.

Etape 0 - 30 min Créer un compte MBED et tester un premier programme

Etape 1 - 45 min Piloter des sorties numériques - LED

Etape 2 - 45 min Acquérir des données numériques - Bouton-poussoirs

Etape 3 - 45 min Mettre en œuvre des interruptions sur des événements externes

Etape 4 - 45 min Utiliser des sorties modulées en largeur d'impulsion (PWM) - LEDs

Etape 5 - 60 min Acquérir des données analogiques - Potentiomètre

Séance 2 / Prise en main de la maquette et déplacements élémentaires

Etape 6 - 60 min Piloter l'intensité des LEDs de la maquette

Etape 7 - 90 min Piloter les moteurs à courant continu

Etape 8 - 60 min Acquérir des données des capteurs de ligne

Etape 9 - 60 min Piloter les phares du robot (NeoPixel)

Séances 3 et 4 - Pilotage de haut niveau

Les deux séances suivantes seront consacrées au pilotage du robot pour lui permettre de suivre une ligne ou/et d'éviter les obstacles qu'il rencontre sur son chemin.

Les étapes possibles sont les suivantes :

Etape 10 - 120 min Définir et tester une première structure de code permettant de piloter les deux moteurs du robot en fonction de la détection des lignes

Etape 11 - 90 min Acquérir les signaux du capteur ultrason

Etape 12 - 90 min Piloter le servomoteur associé au capteur ultrason

Etape 13 - 180 min Améliorer le programme de contrôle du robot

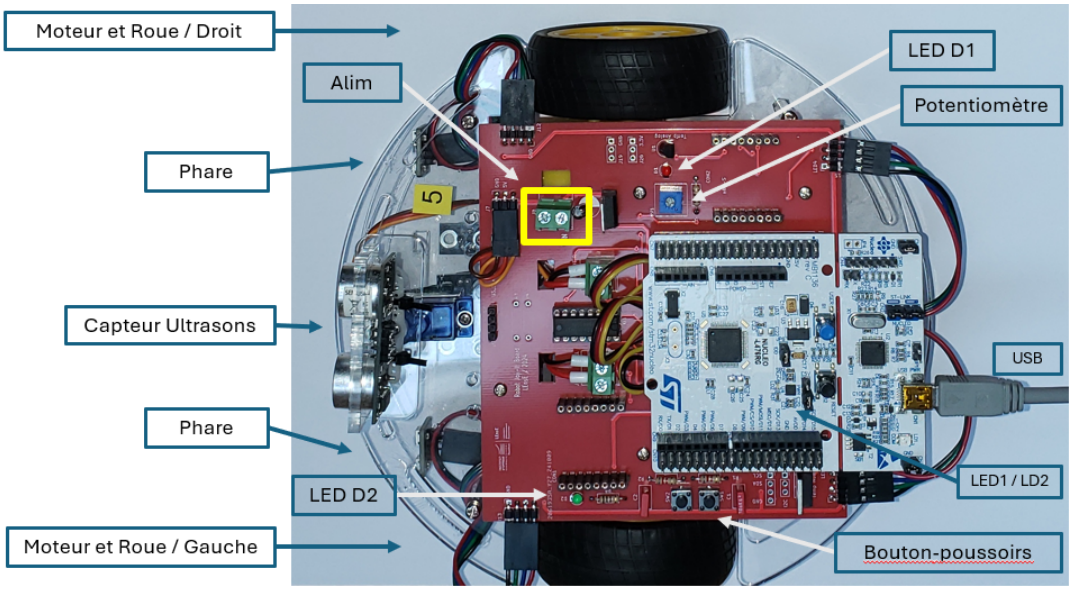
Vous pourrez également ajouter d'autres éléments présents sur la carte : encodeur de vitesse sur les roues, capteurs de température (analogique ou numérique en I2C), accéléromètre (I2C).

Séance 2 / Prise en main de la maquette et déplacements élémentaires

Objectifs de la séance

Cette seconde séance est consacrée à la **prise en main de la maquette** et au développement des **fonctionnalités permettant les déplacements élémentaires** de la plateforme.

Description de la maquette



La maquette est basée sur une plateforme robotique **Joy-It** *Joy-Car* (joy-it.net/en/products/mb-joy-car) constituée :

- de **deux moteurs à courant continu** indépendants, contrôlés par un driver de type L293DN - tension maximale 7 V associés à deux encodeurs de position sur les roues motrices ;
- de **trois capteurs de ligne** (en dessous du robot) ;
- d'un **capteur à ultrason** (obstacles) monté sur un servomoteur ;
- de **4 phares**, composés de 2 LEDs RGB de type WS2812 ;
- d'une **carte de contrôle**, composée :
  - de deux bouton-poussoirs ;
  - de deux LEDs standard ;
  - d'un potentiomètre ;
  - d'une carte Nucléo L476RG.

Un schéma de la carte et le brochage des différents éléments sont donnés en annexe de ce document.

Alimentation électrique

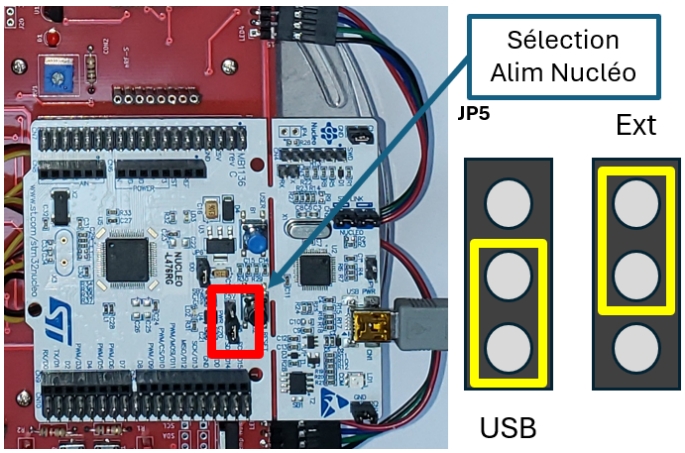
La plateforme robotique est utilisable dans deux configurations différentes :
— **non-autonome**, reliée en USB, lors de la phase de programmation de la carte de commande ;
— **autonome**, sur batterie ou alimentation externe.

Carte Nucléo

La carte Nucléo ainsi que la carte de commande (incluant les phares et les capteurs du robot) peut être **alimentée à l'aide du câble USB** (permettant également la programmation du microcontroleur embarqué sur la carte Nucléo).

Attention ! Dans ce régime de fonctionnement, les moteurs et servomoteur ne sont pas utilisables (car non alimentés).

Le cavalier **JP5** de la carte Nucléo doit être positionné du côté **U5V**.



Attention ! Lors de la programmation de la carte Nucléo, il est préconisé de **ne pas utiliser la partie puissance** !

Utilisation de la partie puissance (motorisation)

La tension maximale admissible par les moteurs est de 7 V !

L'alimentation de cette partie se fait à l'aide du connecteur **J8**. La broche **VIN** correspond à la tension positive (comprise entre 5 et 7V) et l'autre broche est reliée à la masse du système.

Attention ! Il est conseillé de tester la motorisation du robot à l'aide d'une alimentation stabilisée - et surtout protégée en courant ! - avant de passer à l'utilisation totalement autonome sur batterie.

Attention ! Pour utiliser la partie puissance, il est préconisé de passer la **carte Nucléo en utilisation autonome** (sur alimentation externe) et ainsi supprimer le lien par le câble USB.
Pour cela, il faut placer le cavalier **JP5** de la carte Nucléo du côté **E5V** (E = externe).

Brochage

Entrées-Sorties standard

Maquette	**Broche Nucléo**	Type	Description
LED1	PC7	Sortie / PWM	Led active à l'**état bas**
LED2	PB13	Sortie / PWM	Led active à l'**état haut**
SW1	PA11	Entrée	Bouton-poussoir, par défaut état bas
SW2	PA12	Entrée	Bouton-poussoir, par défaut état bas
USERBUTTON	PC13	Entrée	Bouton-poussoir, par défaut état haut
POT_OUT	PC3	Entrée analogique	Potentiomètre

Moteurs

Maquette	**Broche Nucléo**	Type	Description
MOT_EN	PA9	Sortie	Validation des moteurs
MOT_L_1	PB4	Sortie / PWM	Moteur Gauche direction 1
MOT_L_2	PA8	Sortie / PWM	Moteur Gauche direction 2
MOT_R_1	PA0	Sortie / PWM	Moteur Droit direction 1
MOT_R_2	PA1	Sortie / PWM	Moteur Droit direction 2

Phares NeoPixel / SW2812

Maquette	**Broche Nucléo**	Type	Description
DIN_1	PC0	Sortie	Phare avant droit
DIN_2	PA10	Sortie	Phare avant gauche
DIN_3	PC5	Sortie	Phare arrière gauche
DIN_4	PA13	Sortie	Phare arrière droit

Capteurs

Maquette	**Broche Nucléo**	Type	Description
TEMP_OUT	PC2	Entrée analogique	Capteur de température MCP9700
LINE_L	PA7	Entrée	Capteur de ligne Gauche
LINE_C	PB6	Entrée	Capteur de ligne Centre
LINE_R	PA5	Entrée	Capteur de ligne Droit
SPEED_L	PC9	Entrée	Vitesse moteur Gauche
SPEED_R	PC8	Entrée	Vitesse moteur Droit
US_TRIG	PB5	Sortie	Capteur ultrason - Trig
US_ECHO	PB3	Entrée	Capteur ultrason - Echo
SERVO	PB7	Sortie / PWM	Servomoteur du capteur Ultrason

Etape 6 / Piloter l'intensité des LEDs de la maquette

➜ **M** Réaliser un programme qui permet de modifier la luminosité des diodes LED1 et LED2 de la maquette à l'aide des deux bouton-poussoirs SW1 et SW2 (un pour augmenter, l'autre pour diminuer la luminosité). **Votre programme devra utiliser uniquement le principe d'interruption.**

Etape 7 / Piloter les moteurs à courant continu

Un **moteur à courant continu** est un système permettant de **transformer une énergie électrique en mouvement de rotation** continue. Ce type de moteur doit être parcouru par un courant continu.

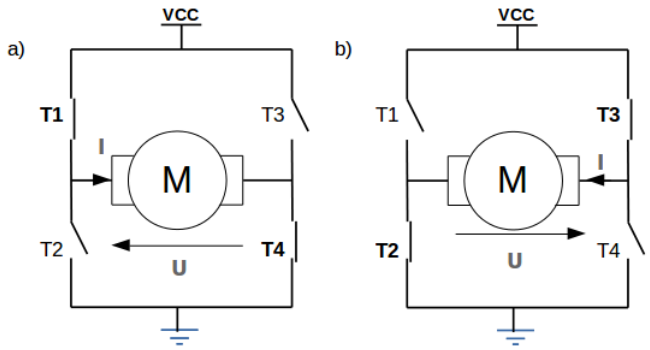
La **vitesse de rotation** d'un tel moteur est liée à la **tension d'alimentation** de ce dernier.

Conversion de puissance

Ces composants de puissance nécessitent souvent des alimentations externes pour fonctionner et ne peuvent pas être directement alimentés par des microcontroleurs (tel que celui présent sur la carte Nucléo). Il est donc indispensable d'ajouter des **étages de conversion de puissance** permettant de passer des signaux de commande (basse puissance) sortant des organes de controle (microcontroleur par exemple) à des signaux de puissance.

Pont en H

Il existe une structure (souvent intégrée dans ce qu'on appelle des drivers) permettant le pilotage d'un moteur dans les deux directions. Cette structure se nomme un **pont en H** et utilise 4 transistors.



Il faut alors piloter en même temps deux des quatre transistors pour laisser passer du courant dans une direction particulière dans le moteur :

— piloter T1 et T4 (figure a) permet de faire tourner le moteur dans un sens ;
— piloter T2 et T3 (figure b) permet de faire tourner le moteur dans l'autre sens.

L293D ou SN754410NE

Le composant **L293D** (ou son cousin **SN754410NE**) intègre deux structures de ce type pour pouvoir piloter 2 moteurs indépendamment dans deux directions (ou 4 moteurs dans une direction).

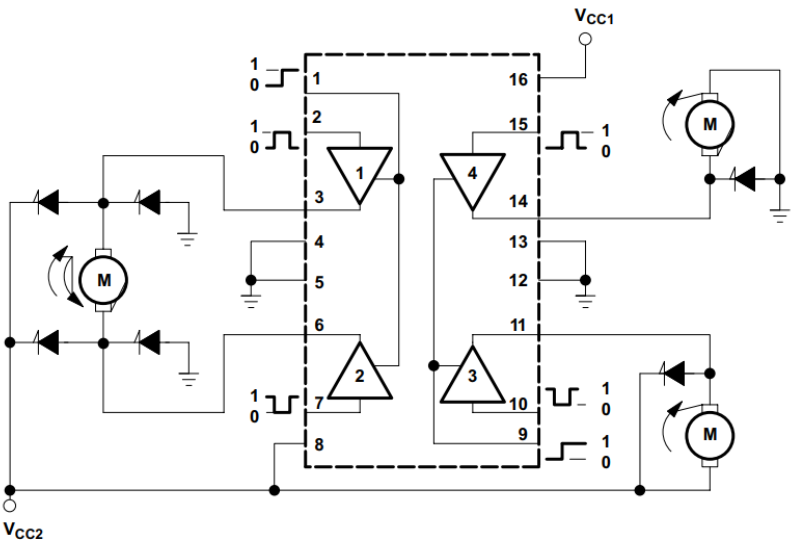


Schéma de cablage du L293D / Documentation Texas Instrument

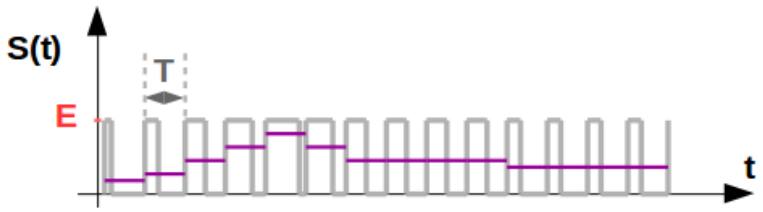
Les **entrées 2 et 7** du composant permettent de faire **tourner le moteur dans un sens ou dans l'autre**. L'**entrée 1** est un **interrupteur général** des modules 1 et 2.

Les signaux des entrées 1, 2 et 7 sont des signaux de faible puissance provenant de la carte Nucléo. Les sorties 3 et 6 sont des sorties de puissance, qui tirent leur énergie de l'alimentation V_{CC2}. Une alimentation de faible puissance (typiquement 5V) - qui doit être indépendante de la source de puissance - doit également être fournie au composant sur V_{CC1}.

Utilisation de signaux modulés en largeur d'impulsion

Les **moteurs à courant continu** se pilotent grâce à des **tensions continues**. Or dans les deux structures vues précédemment, la tension appliquée au moteur ne peut prendre que 3 valeurs : $-V_{CC}$, $0V$ ou V_{CC}, ne laissant alors que 3 réactions possibles du moteur : tourner à une vitesse fixe Ω (dépendant de la valeur de V_{CC}) dans une direction ou dans une autre, ou être à l'arrêt.

En utilisant le **principe de la modulation en largeur d'impulsions** (*PWM* en anglais) de signaux numériques "rapides", il est toutefois possible de contrôler la vitesse de rotation des moteurs à courant continu.



Signal PWM et sa valeur moyenne

En effet, les systèmes mécaniques, que sont les moteurs, ont des **constantes de temps de réaction assez élevées** (de l'ordre de la centaine de millisecondes pour les moteurs de ce robot). En leur appliquant un signal rectangulaire suffisamment rapide, le moteur n'aura pas le temps de réagir aux fluctuations rapides du signal. Seul le **signal moyen** sera alors traduit par le moteur en consigne (système passe-bas).

Travail à réaliser

➜ **M** Créer un nouveau projet dans **Keil Studio Cloud**, en vous basant sur l'exemple ***mbed-os-example-blinky-baremetal*** dans MBED OS 6.

➜ **M** Paramétrer les broches de commande du moteur de gauche en sortie modulée (MOT_L_1 et MOT_L_2). Forcer le signal à un rapport cyclique de 0 dans la fonction d'initialisation.

➜ **M** Paramétrer la broche MOT_EN en sortie standard. Forcer la sortie à 0 pour ce signal dans la fonction d'initialisation.

➜ **M** Ecrire une fonction qui permet de faire tourner le moteur gauche dans une direction à un rapport cyclique donné en argument (entre 0 et 100%).

➜ **M** Ecrire une fonction qui permet de stopper le moteur.

➜ **M** Ecrire une boucle infinie qui permette de tester vos différentes fonctions :
— lancer le moteur à 80% pendant 2 secondes
— stopper le moteur pendant 3 secondes
— lancer le moteur à 50% pendant 4 secondes
— stopper le moteur pendant 1 seconde

➜ **M** Connecter l'alimentation externe sur l'entrée d'alimentation du robot (connecteur J8 - voir partie **Alimentation électrique** de ce document).

➜ **M** Tester votre programme.

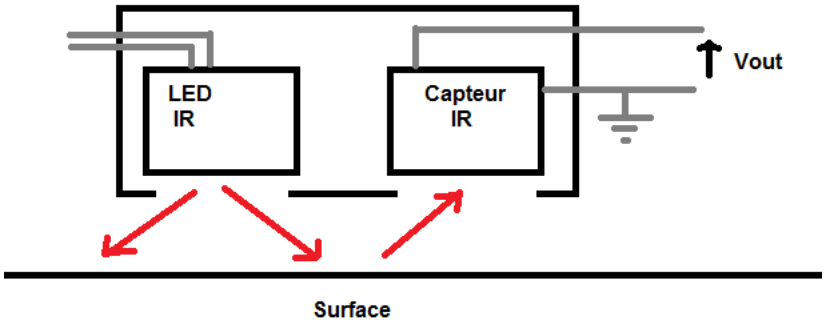
Attention ! Couper l'alimentation externe lorsque vous n'utilisez pas le robot (en particulier lorsque vous modifier et téléverser votre programme dans la carte).

➜ **M** Faire le même travail pour le moteur droit.

➜ **M** Ajouter des fonctions permettant de faire avancer et reculer votre robot. Tester ces fonctions.

Etape 8 / Acquérir des données des capteurs de ligne

Les **capteurs de ligne** utilisés sur cette plateforme sont des **capteurs optiques**. Une **LED infrarouge** émet un signal vers le bas du robot qui est ensuite réfléchie plus ou moins fortement par le matériau présent au sol. Cette réflexion est alors captée par une **photodiode** ou un phototransistor.



Principe du capteur de ligne / gedgeyblog.wordpress.com

Un comparateur est ensuite ajouté en sortie pour transmettre uniquement un signal numérique : '0' si ligne détectée, '1' sinon (ou inversement selon le type de compareteur utilisé). Un potentiomètre de réglage du seuil de détection est présent sur les capteurs. Une LED visible est également présente permettant de visualiser l'état du capteur (cela permet notamment de régler le seuil plus facilement).

Travail à réaliser

➜ **M** Créer un nouveau projet dans **Keil Studio Cloud**, en vous basant sur l'exemple ***mbed-os-example-blinky-baremetal*** dans MBED OS 6.

➜ **M** Paramétrer les broches des **capteurs de ligne** en entrée d'interruption (LINE_L, LINE_C et LINE_R).

➜ **M** Ajouter une fonction d'interruption, exécutée lors de l'activation d'un des capteurs de ligne, permettant de stocker l'état des 3 capteurs dans 3 variables indépendantes (booléennes par exemple) et d'afficher leur valeur sur la console.

➜ **M** Tester votre programme à l'aide d'une feuille blanche contenant une ligne noire de largeur 2 à 3 cm.

Etape 9 / Piloter les phares du robot

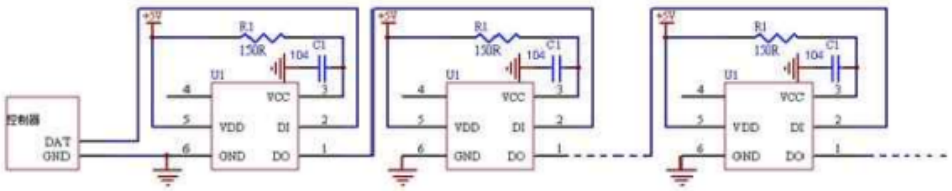
Les phares du robot utilisent des LED trichromes "intelligentes" de type **WS2812**. Ces LEDs peuvent être mises en cascade et se pilotent à l'aide d'une trame numérique. Chaque LED nécessite 24 bits de données (8 bits par couleur - R/G/B).

Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Note: Follow the order of GRB to sent data and the high bit sent at first.

Typical application circuit:



Données et cablage WS2812 / Documentation Adafruit

Bibliothèques

Pour piloter ce type de LED, le LEnsE a modifié deux bibliothèques disponibles dans le fichier *example_WS2812.zip*

— **PixelArray** : permettant de créer des tableaux de pixels R,V,B (rouge, vert et bleu) ;

— **WS2812** : permettant de transmettre les données aux LEDs de type WS2812.

Ce fichier est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Robotique / Exemples MBED pour STM32*.

Importation des bibliothèques

➜ **M** Décompresser le fichier archivé dans un répertoire de votre espace personnel.

Il contient 5 fichiers dont les 4 fichiers des bibliothèques mentionnées précédemment (un fichier .h et un fichier .cpp par bibliothèque).

➜ **M** Créer un nouveau projet dans **Keil Studio Cloud**, en vous basant sur l'exemple ***mbed-os-example-blinky-baremetal*** dans MBED OS 6.

Vous devez importer les deux bibliothèques dans votre projet de la manière suivante :

1. Créer un répertoire ***Libs*** dans votre projet **Keil Studio Cloud**
2. Déplacer les fichiers *PixelArray.h, PixelArray.cpp, WS2812.h* et *WS2812.cpp* dans ce répertoire.

Travail à réaliser

On se propose de tester le code *neoled_control.cpp*. Ce fichier est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Robotique / Exemples MBED pour STM32*.

➜ **M** Tester le code fourni, en ayant au préalable importer les deux bibliothèques précédentes dans votre projet.

➜ **M** Modifier la couleur des deux LEDs présentes sur ce phare.

➜ **M** Ajouter la configuration des 3 autres phares.

➜ **M** Créer une fonction qui permet d'allumer les phares avant, une autre fonction qui permet d'allumer les phares arrière.

➜ **M** Tester vos différentes fonctions.

Séances 3 et 4 / Robot suiveur de ligne autonome

Les deux séances suivantes seront consacrées au pilotage du robot pour lui permettre de suivre une ligne ou/et d'éviter les obstacles qu'il rencontre sur son chemin.

Etape 10 / Définir et tester une première structure de code

On souhaite à présent réaliser le **programme principal** du robot lui permettant de **se déplacer de manière autonome** en suivant une ligne.

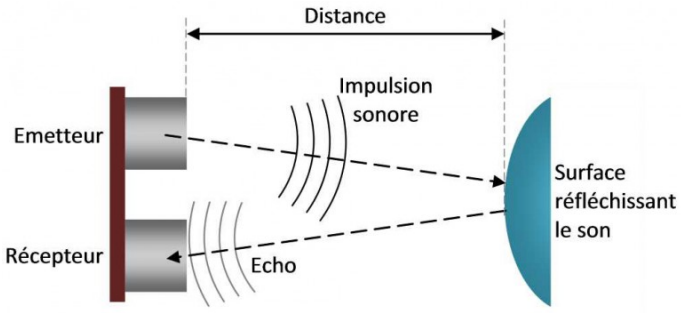
Travail à réaliser

➜ **M** En vous basant sur les différentes fonctions que vous avez écrites précédemment (avancer, reculer, tourner...) et des valeurs des différents capteurs de ligne, réaliser le programme permettant au robot de suivre une ligne.

➜ **M** Tester la robustesse de votre application sur un parcours plus grand.

Etape 11 / Acquérir les signaux du capteur ultrason

La maquette est équipée d'un capteur de distance à ultrasons. Le principe est décrit dans le schéma suivant :



Principe du capteur de distance à ultrasons /
https ://arduino.blaisepascal.fr/capteur-de-distance-a-ultrasons/

Un **émetteur** génère une **impulsion sonore brève** (à une fréquence non audible). Cette onde est **réfléchie** par la surface à détecter. L'onde retour, appelée *écho* est reçue par un **récepteur**. On mesure alors le **temps** entre le moment où l'onde est émise et le moment où l'écho revient - souvent nommé temps de vol. Ces ondes se propageant à la vitesse du son (environ 340m/s), il est alors possible de revenir sur la distance parcourue par cette onde.

Travail à réaliser

On se propose de tester le code *dist_us.cpp*. Ce fichier est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Robotique / Exemples MBED pour STM32*.

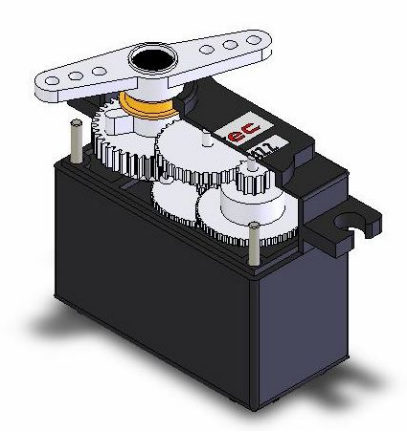
➜ **M** Tester ce code et afficher sur la console Série la durée obtenue en plaçant un objet "plat" devant le robot. Recommencer l'expérience en faisant varier la distance entre l'objet et le robot.

➜ **M** Visualiser les signaux des broches D4 et D3 à l'aide de l'oscilloscope.

➜ **Q** Quelle est la limite de détection de ce capteur ?

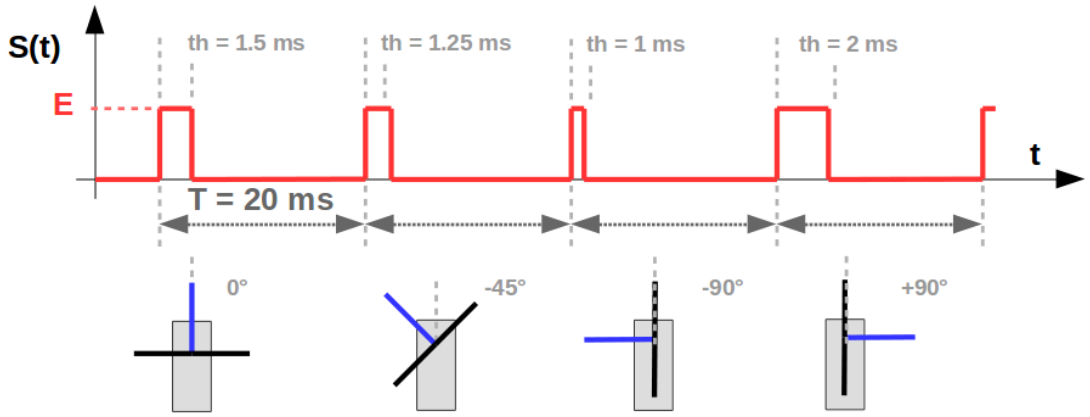
Etape 12 / Piloter le servomoteur associé au capteur ultrason

Un servomoteur est un actionneur qui réalise une rotation d'un angle calibré en fonction d'une commande externe.



Servomoteur - coupe interne (Crédit : redohm.fr)

Le pilotage d'un servomoteur se fait à l'aide d'un **signal modulé en largeur d'impulsions**. Le signal de commande est un **signal rectangulaire** (numérique) de période fixée à 20 ms. C'est ensuite la durée du temps haut qui permet de modifier l'angle de sortie du servomoteur :



Travail à réaliser

On se propose de tester le code *pwm_servo.cpp*. Ce fichier est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Exemples MBED pour STM32*.

➜ **M** Tester le code fourni.

➜ **M** Créer une fonction permettant de positionner le servomoteur à un angle particulier. Tester votre fonction.

Autres fonctionnalités

Acquérir des données de l'accéléromètre (I2C)

Le composant présent sur le robot est un **accéléromètre et magnétomètre** intégrés sur une même puce de silicium. Sa référence est **FXOS8700CQ**. Ce composant est intégré au module *MikroE* **DOF6 - IMU Click**.

Brochage

Ce module fonctionne à l'aide du protocole I2C.

Maquette	**Broche Nucléo**	Type	Description
SDA	PB9	Entrée-Sortie	Signal de données bidirectionnel
SCL	PB8	Sortie	Signal d'horloge
RESET	PC4	Sortie	Reset matériel du composant
Interrupt	PB10	Entrée	Interruption sur réception

Protocole I2C

DESCRIPTION PROTOCOLE et CONNECTIQUES !

ATTENTION ! Les broches utilisées sur la carte Nucléo pour l'I2C ne sont pas celles par défaut. Il est indispensable de préciser les broches SDA et SCL à l'aide des méthodes suivantes :

```
1  Wire.setSDA( PB9 );
2  Wire.setSCL( PB8 );
```

➜ **M** Ouvrir le code *09_accelero.ino* fourni. Compiler ce code et téléverser ce code dans la carte Nucléo.

Ce code contient les fonctions *test_FXOS()* et *read_i2c_buffer()*, ainsi que des définitions des registres internes du composant.

➜ **M**

Configuration

Récupération des données

These registers contain the X-axis, Y-axis, and Z-axis 14-bit left-justified sample data expressed as 2's complement numbers. [NXP Doc p.52 of 113]

Autres fonctionnalités de la carte

La carte de contrôle est également équipée de 2 connecteurs permettant d'accueillir :
— un **accéléromètre** MikroE-6DOF IMU 3 Click ;
— un module de **communication** nRF24L01 ;
Elle est également équipée de **deux capteurs de température** :
— un capteur analogique - MCP9700
— un capteur à sortie numérique - TC74A2

Capteur température numérique / TC74A2

Ce module fonctionne à l'aide du protocole I2C.

Maquette	**Broche Nucléo**	Type	Description
SDA	PB9	Entrée-Sortie	Signal de données bidirectionnel
SCL	PB8	Sortie	Signal d'horloge

Communication nRF24L01

Ce module fonctionne selon le protocole SPI. *Il doit nécessairement être utilisé avec un second module afin de pouvoir transmettre des données entre deux microcontroleurs.*

Maquette	**Broche Nucléo**	Type	Description
SPI			
SCK	PC10	Sortie	Signal d'horloge
MISO	PC11	Entrée	Données entrantes
MOSI	PC12	Sortie	Données sortantes
CS	PA14	Sortie	Sélection du composant
CE	PD2	Sortie	Validation du composant (puissance)
INT	PA15	Entrée	Interruption sur réception

Utilisation de la sortie modulée PB7

```
1  LL_GPIO_SetAFPin_0_7(GPIOB,  GPIO_PIN_7,  GPIO_AF1_TIM2);
```

Traceur Série

```
1     Serial.print(valeur1);
2     Serial.print(",");
3     Serial.print(valeur2);
4     Serial.print(",");
5     Serial.print(valeur3);
6     Serial.print(",");
7     Serial.print(valeur4);
8     Serial.println();
```

Interfaçage Numérique

Travaux Pratiques

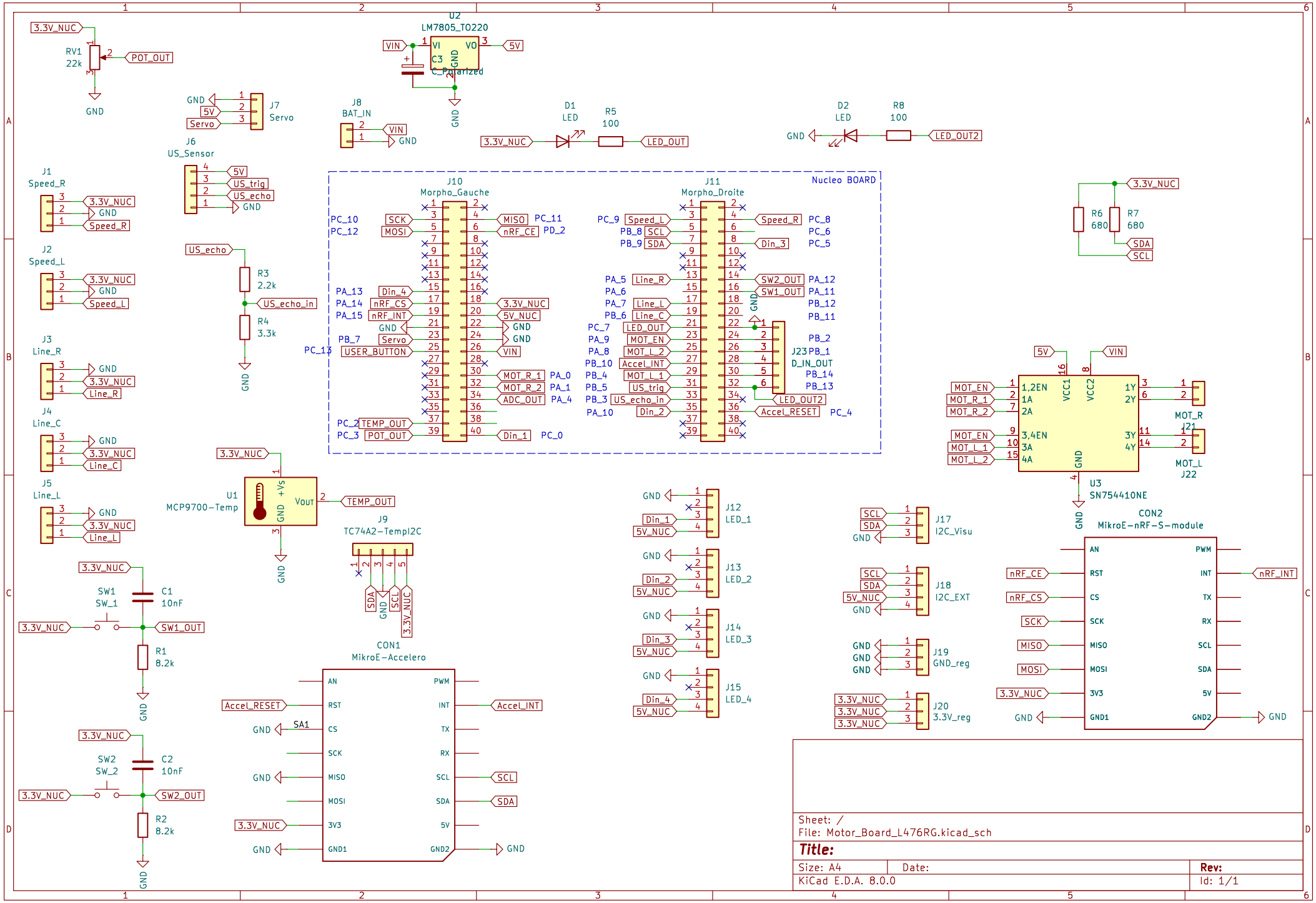
Semestre 6

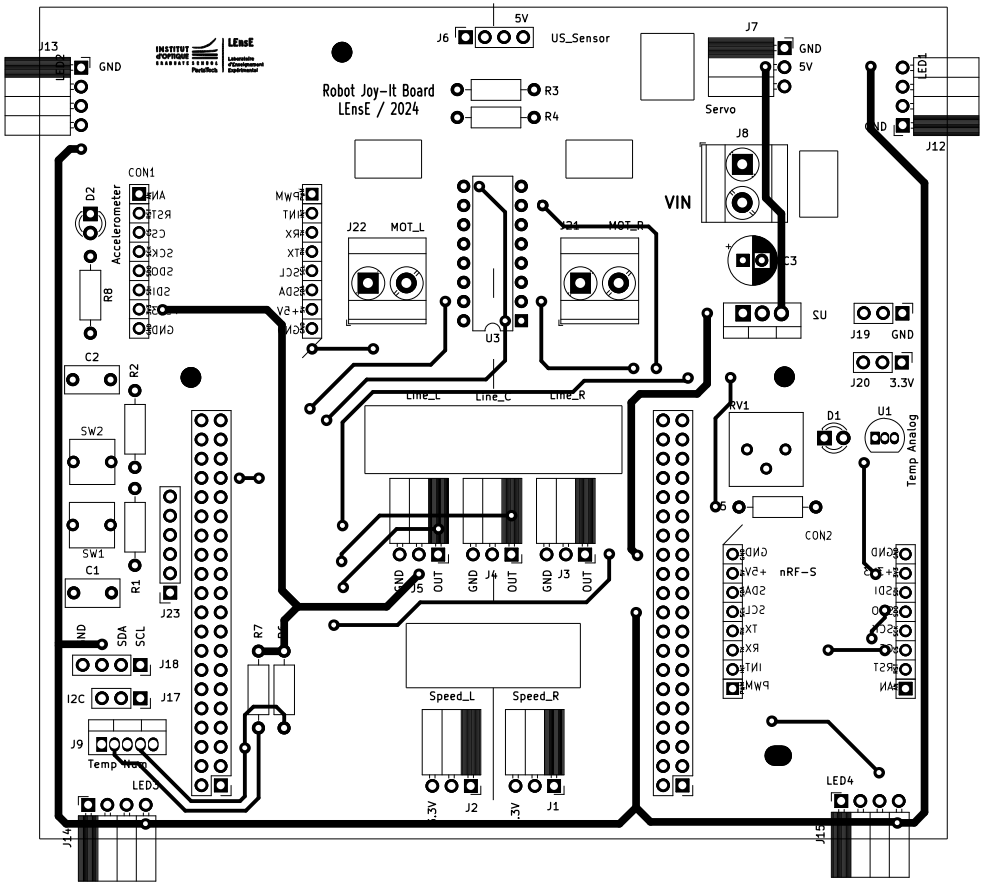
Ressources

Bloc Robot

Liste des ressources

— Schéma de la carte du robot Joy-It Car
— PCB de la carte du robot Joy-It Car


3.3V_NUC
RV1
22k
POT_OUT
GND
U2
LM7805_TO220
VIN
VI
VO
5V
C3
C_Polarized
GND
J7
Servo
GND
5V
Servo
J8
BAT_IN
VIN
GND
D1
LED
R5
100
LED_OUT
D2
LED
R8
100
LED_OUT2
J6
US_Sensor
5V
US_trig
US_echo
GND
J1
Speed_R
3.3V_NUC
GND
Speed_R
J2
Speed_L
3.3V_NUC
GND
Speed_L
J3
Line_R
GND
3.3V_NUC
Line_R
J4
Line_C
GND
3.3V_NUC
Line_C
J5
Line_L
GND
3.3V_NUC
Line_L
US_echo
R3
2.2k
US_echo_in
R4
3.3k
GND
Nucleo BOARD
J10
Morpho_Gauche
PC_10
PC_12
SCK
MOSI
MISO
PC_11
nRF_CE
PD_2
PA_13
PA_14
PA_15
Din_4
nRF_CS
nRF_INT
3.3V_NUC
5V_NUC
GND
PB_7
Servo
USER_BUTTON
VIN
PC_13
MOT_R_1
MOT_R_2
ADC_OUT
PA_0
PA_1
PA_4
PC_2
PC_3
TEMP_OUT
POT_OUT
Din_1
PC_0
J11
Morpho_Droite
PC_9
Speed_L
PB_8
SCL
PB_9
SDA
Speed_R
Din_3
PC_8
PC_6
PC_5
PA_5
Line_R
PA_6
PA_7
Line_L
PB_6
Line_C
PC_7
LED_OUT
PA_9
MOT_EN
PA_8
MOT_L_2
PB_10
Accel_INT
PB_4
MOT_L_1
PB_5
US_trig
PB_3
US_echo_in
PA_10
Din_2
SW2_OUT
SW1_OUT
PA_12
PA_11
PB_12
PB_11
PB_2
PB_1
D_IN_OUT
PB_14
PB_13
LED_OUT2
Accel_RESET
PC_4
J23
R6
680
R7
680
SDA
SCL
U1
MCP9700-Temp
+Vs
VOUT
TEMP_OUT
J9
TC74A2-TempI2C
SDA
GND
SCL
3.3V_NUC
U3
SN754410NE
5V
VIN
MOT_EN
MOT_R_1
MOT_R_2
1,2EN
1A
2A
VCC1
VCC2
1Y
2Y
MOT_R
J21
3,4EN
3A
4A
3Y
4Y
MOT_L
J22
CON2
MikroE-nRF-S-module
AN
RST
CS
SCK
MISO
MOSI
3V3
GND1
PWM
INT
TX
RX
SCL
SDA
5V
GND2
nRF_INT
SW1
SW_1
C1
10nF
SW1_OUT
R1
8.2k
SW2
SW_2
C2
10nF
SW2_OUT
R2
8.2k
CON1
MikroE-Accelero
Accel_RESET
SA1
Accel_INT
J12
LED_1
J13
LED_2
J14
LED_3
J15
LED_4
J17
I2C_Visu
J18
I2C_EXT
J19
GND_reg
J20
3.3V_reg
Sheet: /
File: Motor_Board_L476RG.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.0
Id: 1/1


Robot Joy-It Board
LEnsE / 2024
VIN
Servo
US_Sensor
Accelerometer
Temp Analog
Temp Num
nRF-S
Line_L
Line_C
Line_R
Speed_L
Speed_R
MOT_L
MOT_R
J13
J12
J8
J6
J7
J22
J21
U3
C3
R3
R4
R8
C2
R2
SW2
SW1
C1
R1
J23
J18
J17
J9
J14
J15
J5
J4
J3
J2
J1
J19
J20
RV1
D1
U1
D2
LED3
LED4
LED1
CON1
CON2
R7
GND
5V
3.3V
SDA
SCL
I2C