EirLab

Atelier - Conception d'un robot différentiel pilotable

Ateliers destinés aux stagiaires de seconde
ENSEIRB-MATMECA - CEA

Juin 2025

BORDEAUX INP Enseirb-Matmeca

cea

Auteurs :
Mattéo BINET
Bastien TRAN-RUESCHE

Tuteur :
Thierry TARIS

Sommaire

1 Description de l'atelier

1.1 Format

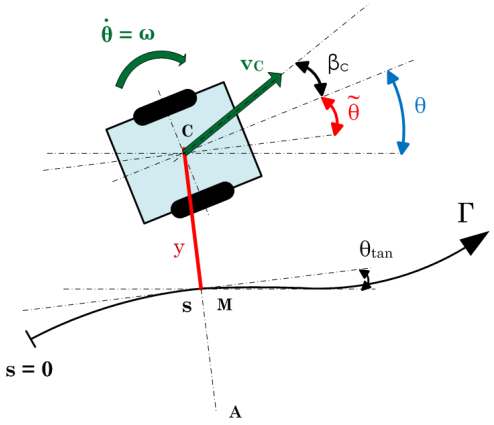
L'atelier se déroule à EirLab (ENSEIRB-MATMECA) et dure 3 heures. Il se découpe en trois parties :

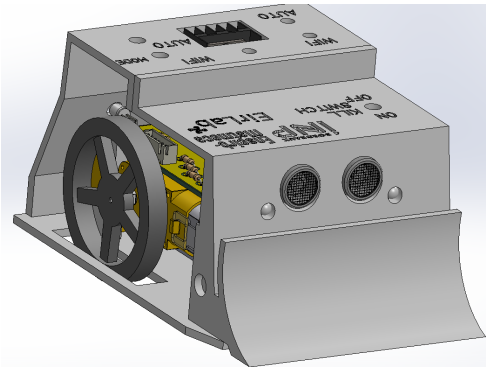
1. Présentation du robot à réaliser
2. Fabrication d'une partie du châssis à la découpeuse laser
3. Programmation du microcontrôleur

1.2 Cahier des charges

L'objectif de cet atelier est l'introduction à la robotique. Pour cela, les élèves de seconde participeront à plusieurs étapes de fabrication du robot. Pour gagner du temps, certaines pièces du châssis seront déjà découpées, la carte d'interconnexion préassemblée et le programme partiellement préparé.
Après la présentation du robot et de son fonctionnement, l'élève sera sollicité pour dessiner et découper la plaque de support du robot, pour assembler ce dernier (soudure et montage), ainsi que pour écrire la partie du programme correspondant au comportement « autonome » du robot.
Ainsi, un robot fonctionnel doit pouvoir être obtenu à la fin de l'atelier.

2 Présentation du robot





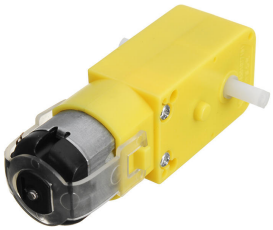
Le robot possède deux roues. Chaque roue est entraînée par un moteur que l'on peut contrôler de manière indépendante. Un tel robot peut effectuer plusieurs mouvements distincts :

1. Si les deux roues tournent dans le même sens et à la même vitesse, le robot va tout droit,
2. Si une des deux roues tourne plus vite, le robot tourne un peu,
3. Si les deux roues n'ont pas le même sens de rotation, le robot tourne sur lui-même.

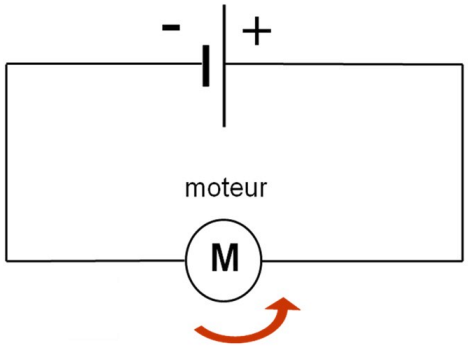
Cela veut dire que pour faire se mouvoir le robot comme on le veut, il faut contrôler le sens et la vitesse de la rotation de chaque moteur.

Le contrôle des moteurs est réalisé à l'aide : d'un microcontrôleur et d'un driver de moteur.

2.1 Les moteurs DC



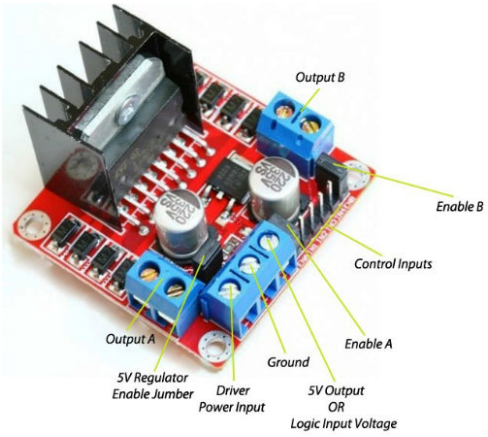
(a) Moteur DC



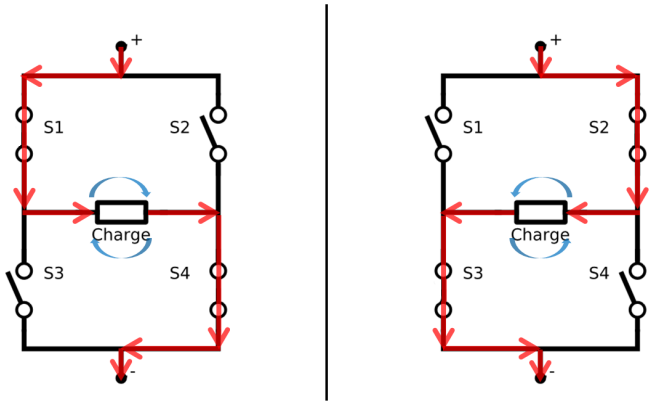
(b) Schéma électrique de branchement moteur

Direct Current = Courant Continu
Un moteur à courant continu est facilement contrôlable. Il suffit de le faire traverser par un courant (en appliquant une tension à ses bornes). En fonction de l'intensité et du sens du courant, il est possible de contrôler la vitesse et le sens de la rotation.

2.2 Le driver du moteur



(a) Driver L298



(b) Schéma électrique du pont en H

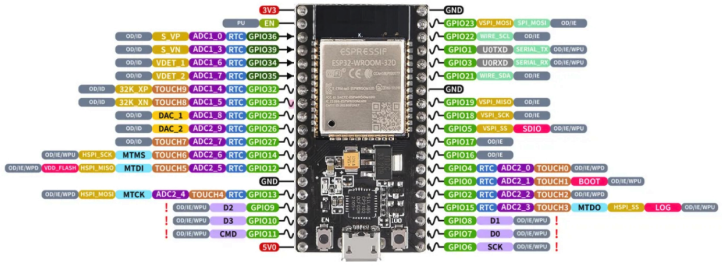
Pour fournir la tension contrôlée aux moteurs ($\approx 12V$), un composant de puissance est nécessaire car l'Arduino ne fournit pas une tension suffisante ($\approx 5V$). Ce composant est appelé « driver » (peut se traduire par entraîneur).

La référence utilisée est le pont en H L298. Sans entrer dans les détails, il est commandé par un signal analogique donné par l'Arduino (entre 0 et 5V), et fournit aux moteurs une

tension proportionnelle à ce signal.

Le driver permet ainsi de commander les moteurs de manière précise, mais avec puissance.

2.3 Le microcontrôleur (Arduino)



(a) Broches de l'ESP32

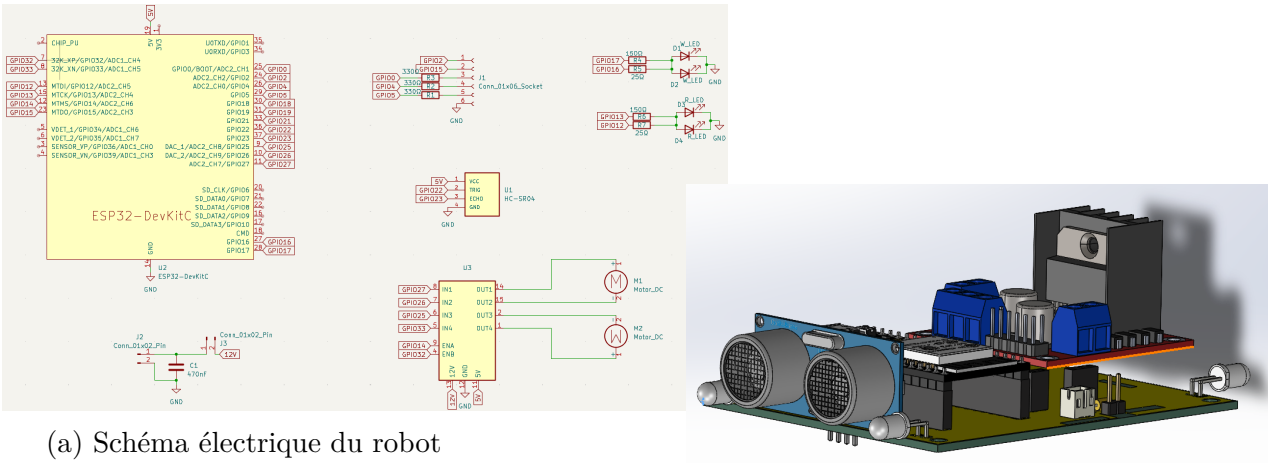
Un microcontrôleur est une carte programmable. En fonction du code qu'on lui donne, elle peut lire ou appliquer des tensions à ses bornes au cours du temps.

La carte utilisée est une ESP32, mais elle possède le même comportement qu'une carte Arduino (plus répandue et plus documentée).
Son rôle dans le robot est :

1. La mesure des entrées comme les boutons, le capteur à ultrasons...
2. Le choix des actions à réaliser en fonction de l'état des entrées
3. L'application des signaux de contrôle à ses sorties, vers le driver et les lumières par exemple.

Toutes ces étapes doivent être rigoureusement définies dans le programme donné au préalable au microcontrôleur. Si on a l'impression que le robot est autonome, c'est parce que son code est bien réalisé.

2.4 La carte d'interconnexion



(a) Schéma électrique du robot

Pour ce projet, une carte électronique a été préparée. Elle sert à relier les différents composants entre eux. Par exemple, six broches de l'ESP32 sont reliées aux six entrées de contrôle du driver. De manière similaire, l'alimentation 5V est partagée entre le capteur à ultrasons, la carte Arduino et le driver...
Les composants électroniques du robot sont :

1. le microcontrôleur ESP32,
2. les interrupteurs (3),
3. les lumières (7),
4. le capteur à ultrasons,
5. le driver moteur,
6. les moteurs (2),
7. la prise pour la batterie

3 Partie Mécanique

3.1 Objectif

La partie mécanique doit permettre aux stagiaires de fabriquer une partie du châssis du robot (la plaque inférieure) et de souder les moteurs à la carte d'interconnexion. Ils seront aussi amenés à assembler certaines parties du robot.

3.2 Fabrication de la plaque

Le logiciel utilisé n'importe pas beaucoup. Le seul point important est que le dessin final doit pouvoir être obtenu au format .svg .
Le dessin de la carte se fait suivant les cotes suivantes :
Considérations lors du dessin :

— Ne déssiner que la moitiée de la plaque. L'autre moitée s'obtient par symétrie,
— Placer les trous d'un même élement (supports moteurs, supports carte, trous des roues...) les un par rapport aux autres
— Placer les différents éléments par rapport à une même référence. On peut choisir la ligne de derrière et une ligne du côté (il faut donc les placer en premier).

3.3 Soudure

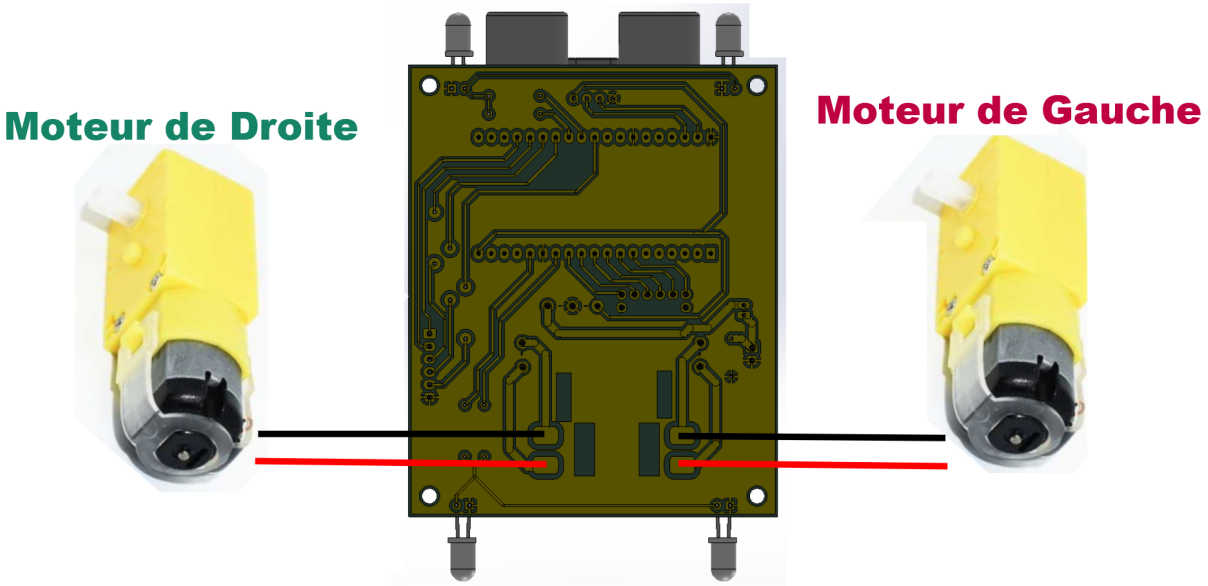


FIGURE 8 – Schéma de câblage des moteurs

Il faut relier les moteurs à la carte d'interconnexion. Chaque moteur possède deux fils : il y a donc 4 fils à souder sur la carte.

/!\ Pour la sécurité /!\ (gardez vos mains et vos yeux fonctionnels)

1. Porter des **lunettes de sécurité**
2. Souder sous supervision mais **seul sur la table**
3. Soit vous utilisez activement le fer, soit il est **rangé dans son socle**
4. Lors de la soudure, **garder le fer loin** des surfaces en plastiques et de votre autre main.
5. Les surfaces sont encore **brulantes** pendant **plusieures minutes**
6. Les troisièmes mains sont ajustables. Pas besoin de se mettre dans des positions **incomfortables et dangeureuses.**

Conseil pour une soudure de compétition :

— régler son fer entre 350° et 380°.

— nétoyr puis étamer son fer à chaque utilisation

— étamer les deux surfaces individuellement. Ensuite, réunir les deux surfaces ainsi que le fer. Il faut bien attendre que tout l'étain soit fondu avant d'enlever le fer.

— L'étain ne se solidifie pas instantanément, il faut garder les surfaces à souder immobiles.

— Ne pas inverser + et - ; ne pas iverser moteur droit et moteur gauche.

4 Partie Programmation

4.1 Objectif

L'objectif de la partie programmation est que le stagiaire programme le comportement autonome du robot.
La partie contrôle avec son téléphone à distance ainsi que les fonctions pour le déplacement du robot (avancer, aller à droite...) sont déjà programmées. Le stagiaire pourra les utiliser.

Des aides à la programmation sont aussi données plus loin dans le document.

4.2 Etapes de programmation

Dans cette partie, le stagiaire doit programmer le robot. Vu le temps accordé à l'atelier, une partie du code est déjà réalisée.

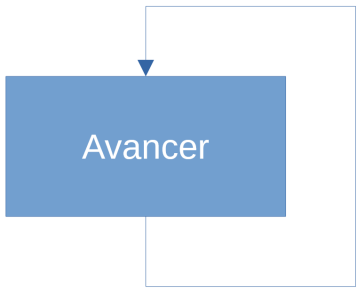
Documents fournis par les encadrants :

1. La librairie *pami_lib.h* du robot. On y trouve les fonctions pour faire bouger le robot comme avancer, reculer...
2. Le code *main.c* prérempli. L'import des librairies, la définition des pins et la fonction `setup()` sont déjà réalisés.

Travail demandé au stagiaire :

1. Réaliser la fonction `loop()`. Le comportement attendu est décrit ci-dessous dans les différentes étapes.

4.2.1 Étape 1 - Faire avancer le robot

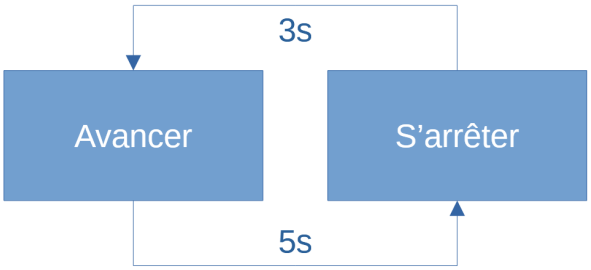


Pour faire avancer le robot, il faut appeler la fonction `avancer()`. On peut dire 3 choses sur cette fonction :

1. Elle est non bloquante. Son exécution est rapide et permet au programme de continuer à défiler. (contre-exemple : `delay()`).
2. Elle change l'état du robot. Tant qu'une instruction contraire n'est pas donnée, le robot continue à avancer.
3. L'appeler une seconde fois ne change rien.

À faire : Dans la `loop()`, écrire l'instruction `avancer();` . Ensuite, téléverser.

4.2.2 Étape 2 - Faire arrêter le robot

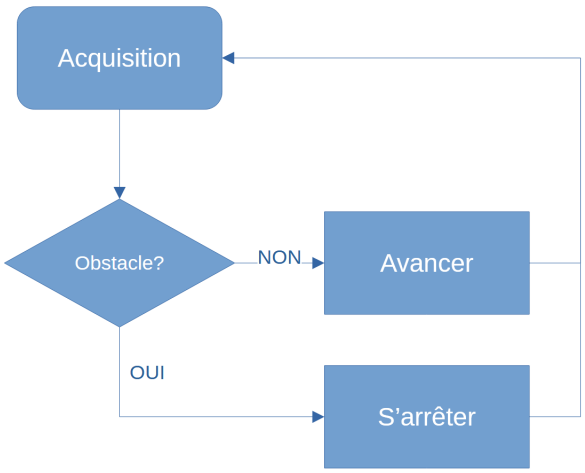


L'objectif est de faire avancer et arrêter le robot. La fonction d'attente est `delay()`. La fonction d'arrêt est `stop()`.
Quelques précisions sur la fonction `delay()` :

1. Elle est bloquante. Son appel va stopper la lecture du programme pendant la durée spécifiée.
2. Aucun changement n'est effectué. Si le robot avance avant l'appel de `delay()`, il continuera d'avancer pendant toute la durée d'attente.

À faire : Modifier le programme pour que le robot : avance pendant 5 secondes, s'arrête pendant 3 secondes, et recommence en boucle...

4.2.3 Étape 3 - Utiliser le capteur à ultrasons

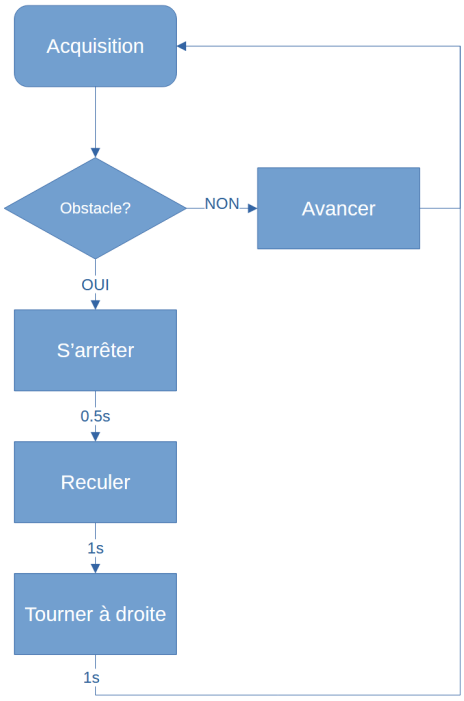


La fonction `mesurerDistance()` renvoie la distance en cm devant le capteur. On veut que le robot s'arrête quand il voit un obstacle devant lui. Il faut donc :

1. Récupérer la distance mesurée
2. La comparer à une valeur de référence (15 cm par exemple)
3. Choisir s'il faut avancer ou s'arrêter

À faire : Écrire dans la `loop()` un programme qui fait avancer le robot tant qu'un obstacle n'est pas détecté.

4.2.4 Étape 4 - Manœuvre d'évitement (Bonus)



On propose maintenant en bonus d'effectuer une manœuvre d'évitement par le robot pour ne pas qu'il reste bloqué.

Bonus : Modifier le programme pour que le robot change de direction quand il détecte un obstacle.

4.3 Aides à la programmation

4.3.1 Variables et fonctions

- `int a = 3;` : entier
- `float b = 3.14;` : nombre décimal
- `b = a + b;` : modification de variable
- `maFonction(3);` : appel de fonction
- `float c = maFonction(a);` : affectation du résultat d'une fonction

/!\ Il faut mettre des points-virgules pour signaler la fin de l'instruction. Exemple : `int a = 3;`

4.3.2 Conditions et boucles

Les tests : `a == b`, `a < b`, `a <= b`, `a > b`, `a == HIGH`...
Les boucles :

- `if(condition){ instruction1; instruction2; ... }` : si
- `if(condition){ } else{ }` : si ... sinon
- `if(condition){ } else if(condition2){ } else{ }` : si ... sinon si ... sinon

— `while(condition){ }` : tant que

— `for(initialisation; condition; incrment){ }` : pour

4.3.3 Fonctions propres à Arduino

— `setup()` : fonction automatiquement appelée au démarrage de la carte

— `loop()` : fonction automatiquement appelée en boucle

— `pinMode(pin, mode);` : fonction pour configurer une broche

— `digitalRead(pin);` : fonction pour lire la valeur d'une broche. Renvoie `LOW` ou `HIGH`

— `digitalWrite(pin, value);` : fonction pour écrire `LOW` ou `HIGH` sur une broche

Les fonctions `setup()` et `loop()` sont appelées automatiquement par la carte Arduino. Elles doivent être définies par l'utilisateur avec le type `void`, car elles ne renvoient rien. Exemple : `void setup(){ ... }`. On utilise la même structure pour définir ses propres fonctions.
En revanche, les fonctions comme `pinMode` sont déjà définies ailleurs. L'utilisateur peut les appeler s'il le souhaite.

/!\ Il faut mettre des points-virgules pour signaler la fin de l'instruction. Exemple : `pinMode(12, INPUT);`

4.3.4 Fonctions de la librairie PAMI

— `beginPami();` : initialise le PAMI

— `avancer();` : pour avancer

— `reculer();` : pour reculer

— `droite();` : le PAMI tourne sur lui-même (sens horaire)

— `gauche();` : le PAMI tourne sur lui-même (sens antihoraire)

— `stop();` : pour s'arrêter

— `mesurerDistance();` : retourne la distance en cm mesurée par le capteur à ultrasons

— `bluetoothControl();` : actualise le comportement du robot en fonction des commandes Bluetooth

/!\ Il faut mettre des points-virgules pour signaler la fin de l'instruction. Exemple : `avancer();`

4.3.5 Autres

— Attente en ms : `delay(ms_a_attendre);`

— Commentaire : `// reste de la ligne en commentaire`

— Inclure une librairie : `#include "nomLibrairie.h"`

— Macro : `#define MA_MACRO 12`

4.3.6 Un exemple pour illustrer

Le programme suivant scrute l'interrupteur sur la broche 13. S'il est activé, alors c'est le capteur à ultrasons qui est scruté. Si la distance devant le capteur est supérieure à 15 cm, alors le robot avance. Si une des deux conditions n'est pas remplie, le robot est arrêté.

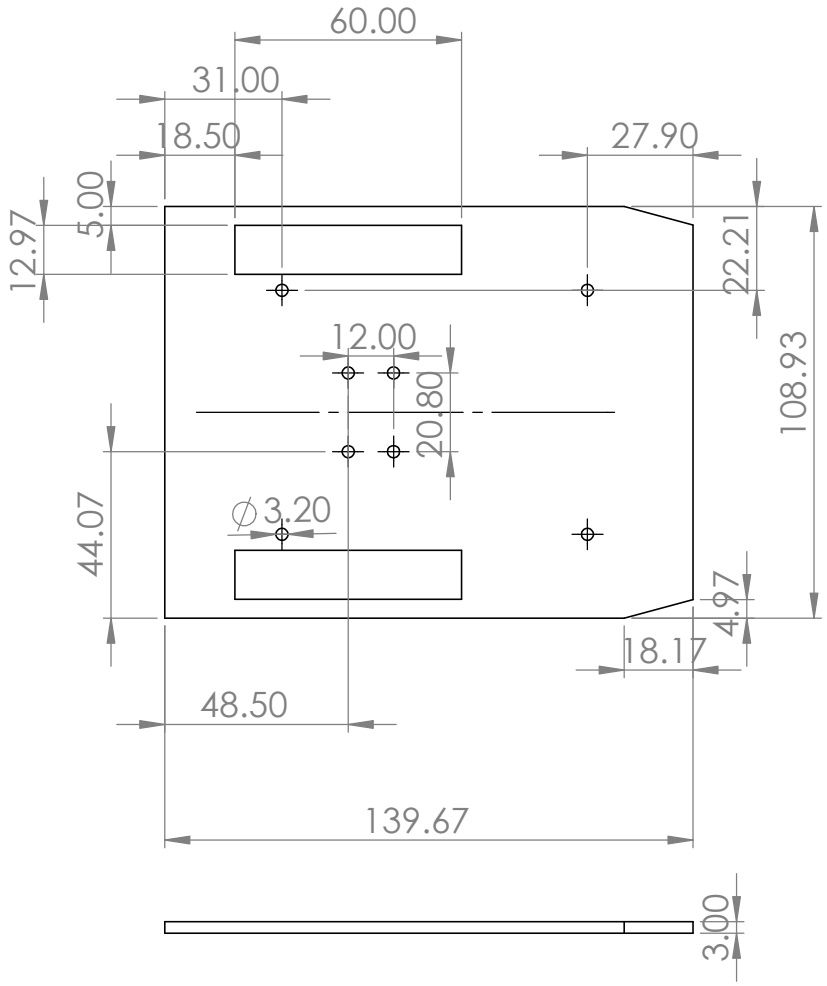
Listing 1 – Exemple C++

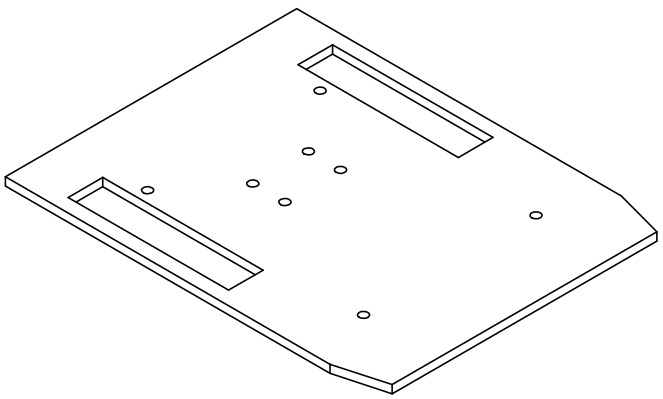
```
#include "pami_lib.h"

#define SWITCH_PIN 13

void setup(){
    beginPami();
    pinMode(SWITCH_PIN, INPUT);
}

void loop(){
    if(digitalRead(SWITCH_PIN) == HIGH){
        int distance = mesurerDistance();
        if(distance >= 15){
            avancer();
        } else {
            stop();
        }
    }
    else{
        stop();
    }
}
```





UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN						châssis robot différentiel	
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO.	A4
				PMMA		plaque_bas	
				WEIGHT:		SCALE:1:2	SHEET 1 OF 1