



Comment concevoir un robot qui peut se déplacer seul dans un espace clos ?

# PILOMO 1.0

Zana  
DIAMOUTENE

Etienne  
RUAULT





## De quoi est capable notre robot aujourd'hui ?

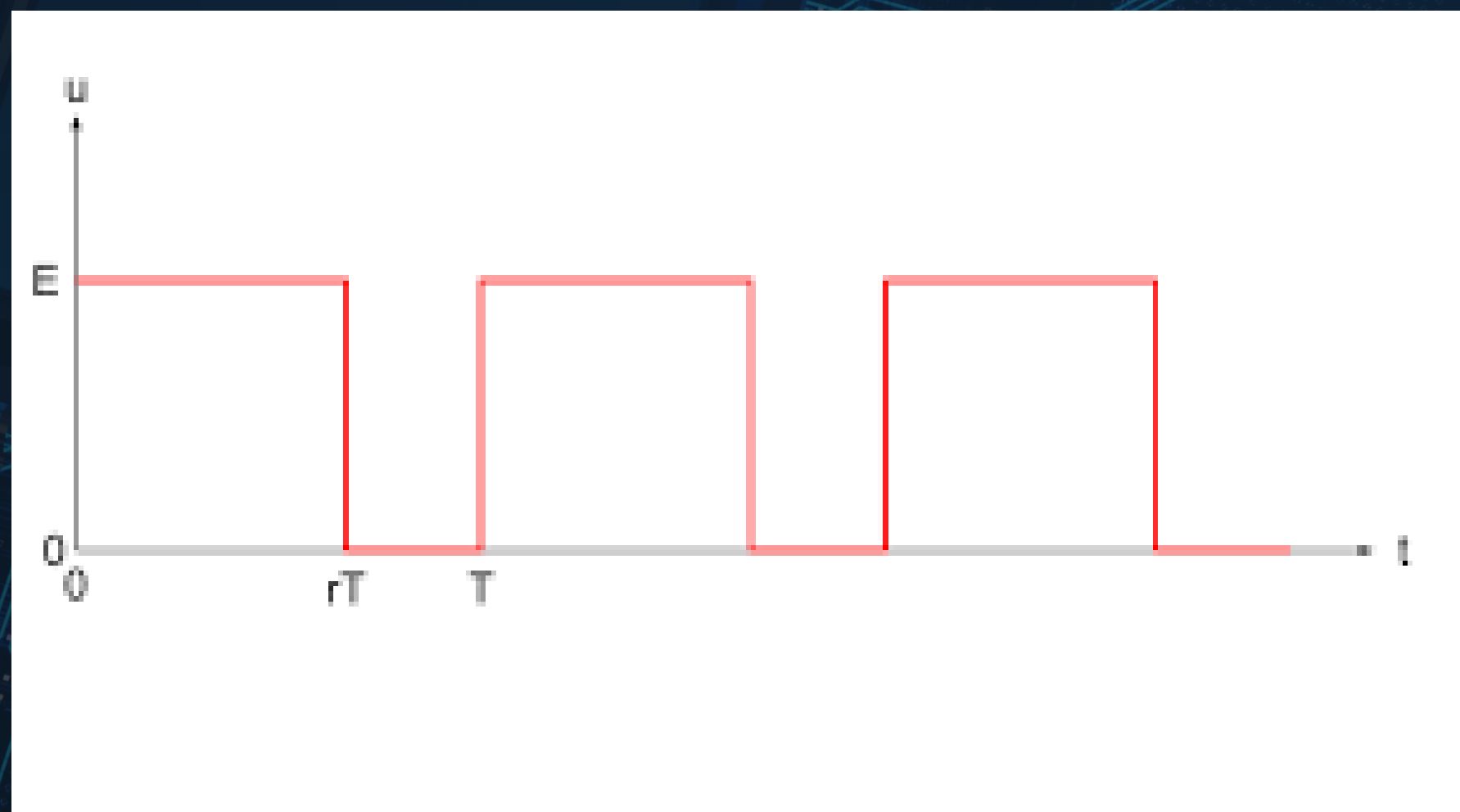
- Déplacement dans un espace entouré de bordures
- Condition d'arrêt : contact capteur moustache / 20s depuis le départ
- Suivre une paroi à 40 cm
- Affichage LCD
- Biélette



# Étude technique

Moteurs à courant continu

Signal MLI ou PWM





## Recherche des composants

La diode à roue libre

	1N4148	1N4001	1N4007
$I_{Fmax}$	200mA	1A	1A
$V_{RRM}$		50	1000
choix	impossible	possible	possible





## Recherche des composants

### Transistor

Parmi les transistors disponibles, nous avons des transistors Bipolaire NPN, Bipolaire PNP, MOSFET canal N et MOSFET canal P. Les transistors Bipolaire NPN et les transistors MOSFET canal N seront fermés pour un niveau haut du signal MLI\_MG de commande, ce qui correspond à notre besoin. Un autre paramètre va nous permettre d'effectuer le choix des transistors possibles, c'est le courant maximum. Ce courant maximum est identique au courant maximum du moteur. En tenant compte de ces informations, choisissez dans la liste fournie (voir premier document) les transistors compatibles avec notre besoin.

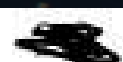
	2N2222	BS170	BS250	BD238	TIP31C	BUZ11
Type	Bipolaire NPN	MOSFET N	MOSFET P	Bipolaire PNP	Bipolaire NPN	MOSFET N
I <sub>max</sub>	800mA	500mA			3A	30A
Choix	Impossible	Impossible	Impossible	Impossible	possible	possible






## Recherche des composants

### Regulateur de tension

 Dans un premier temps on va débiter l'étude avec un régulateur délivrant une tension de +5v (Il n'existe pas de régulateur de +5,6v). Parmi les 5 références de régulateur disponibles, choisissez celui qui correspond au besoin (tension et courant).

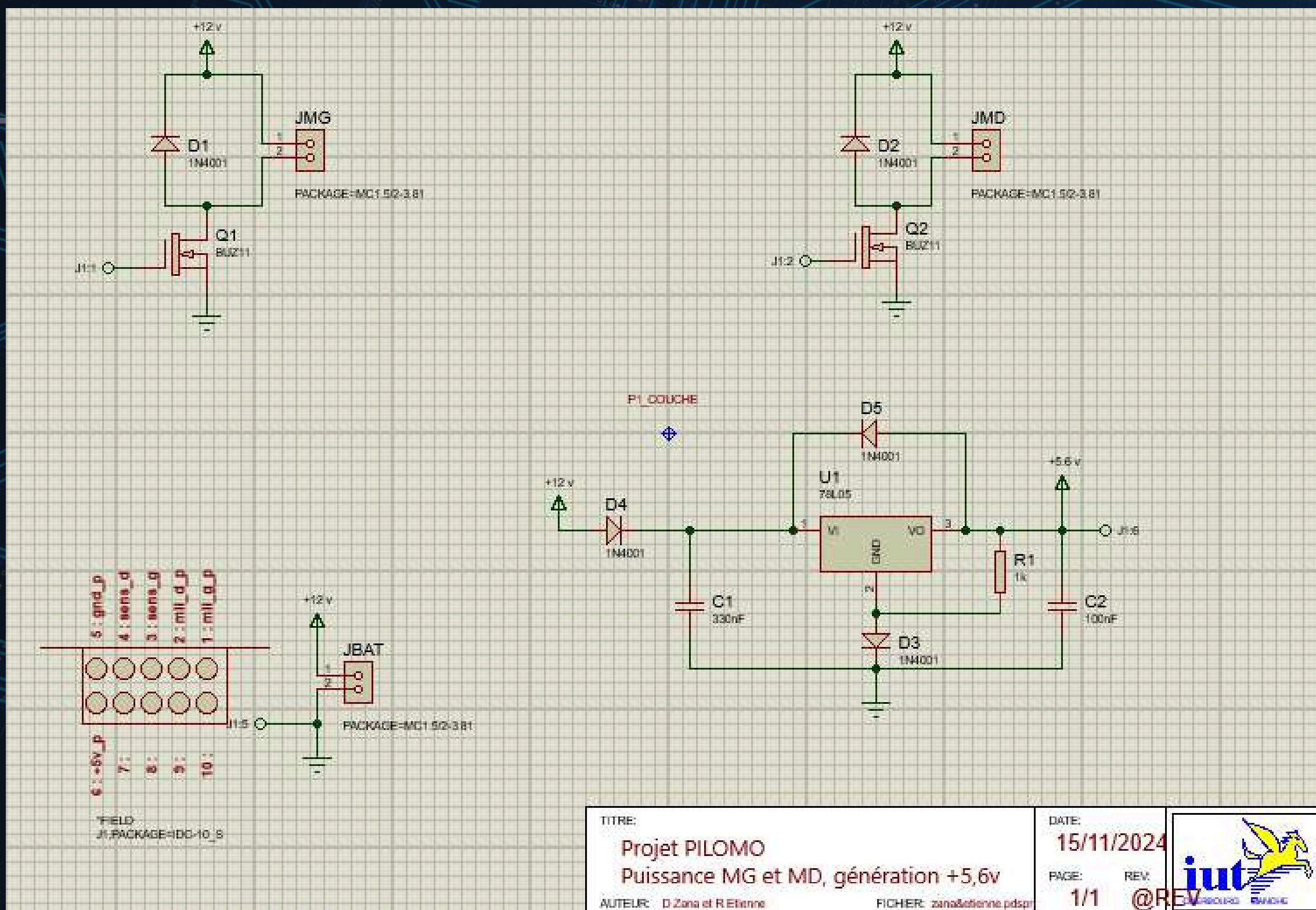
tension de sortie	7805 5v	7809 9v	78L05 5v	78L12 12	LM317 reglable
courant de sortie	1A		100mA		
choix	possible	impossible	possible	impossible	pas souhaité

 Afin d'obtenir les +5,6v en sortie du régulateur 5v, on va « tromper » ce dernier en plaçant sa broche de masse à 0,6v, ainsi on aura bien  $5 + 0,6 = 5,6v$ . Pour ce faire on va utiliser une diode de faible puissance, on sait que la tension à ses bornes lorsqu'elle est montée en directe est de +0,6v. Pour la polariser en directe, on branchera en série une résistance, l'autre extrémité de cette résistance sera connectée à la sortie du régulateur. On aura donc une tension de +5v à ses bornes.



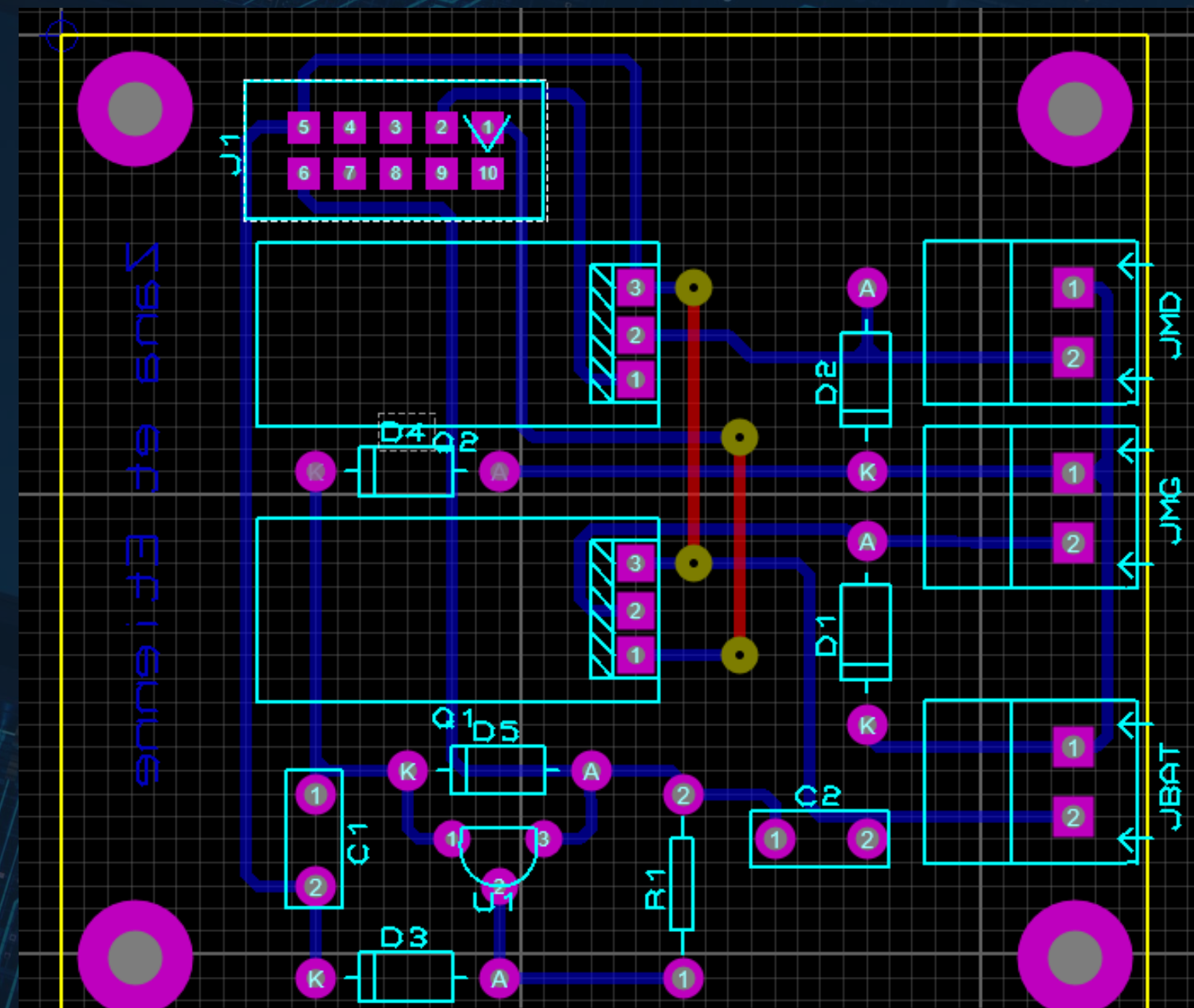
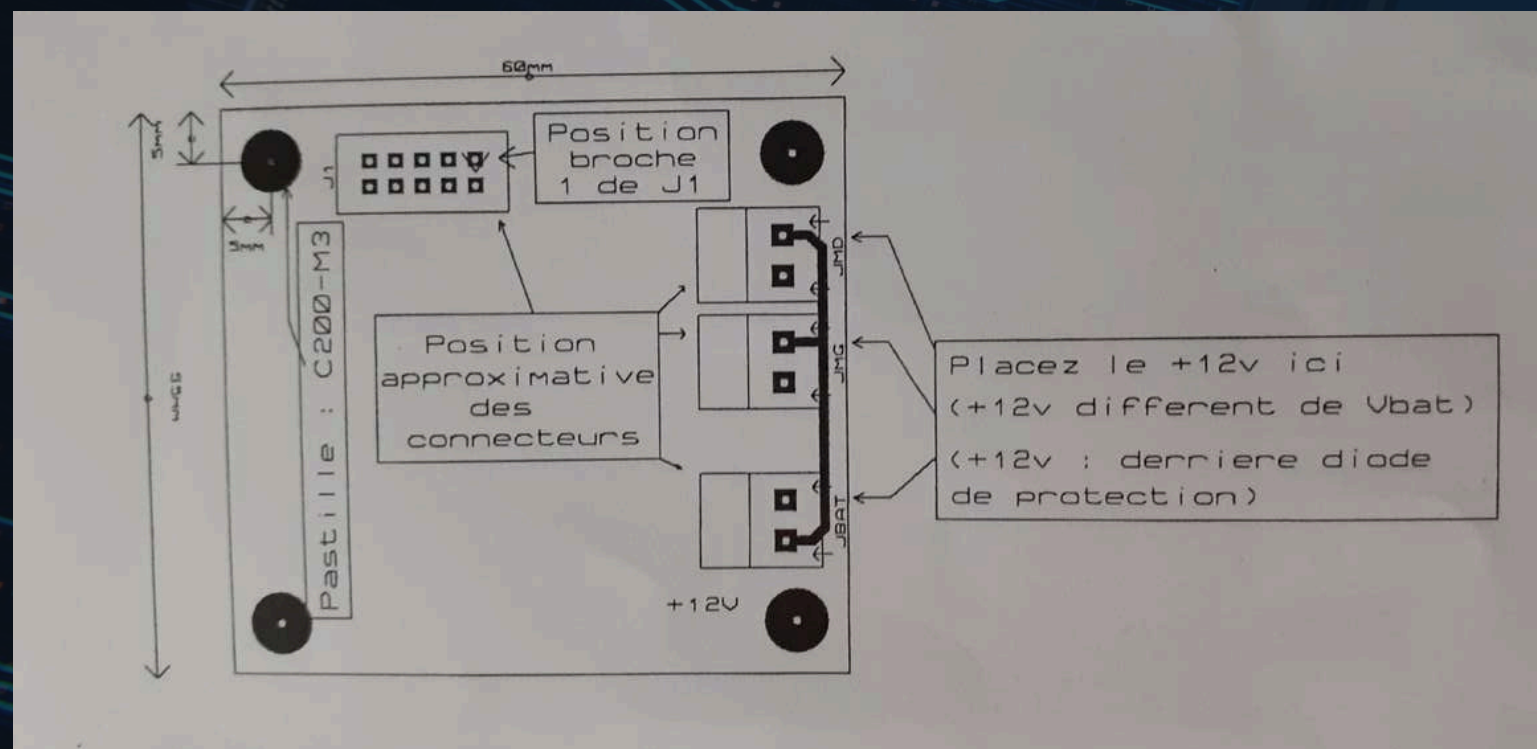


# Saisie de schéma



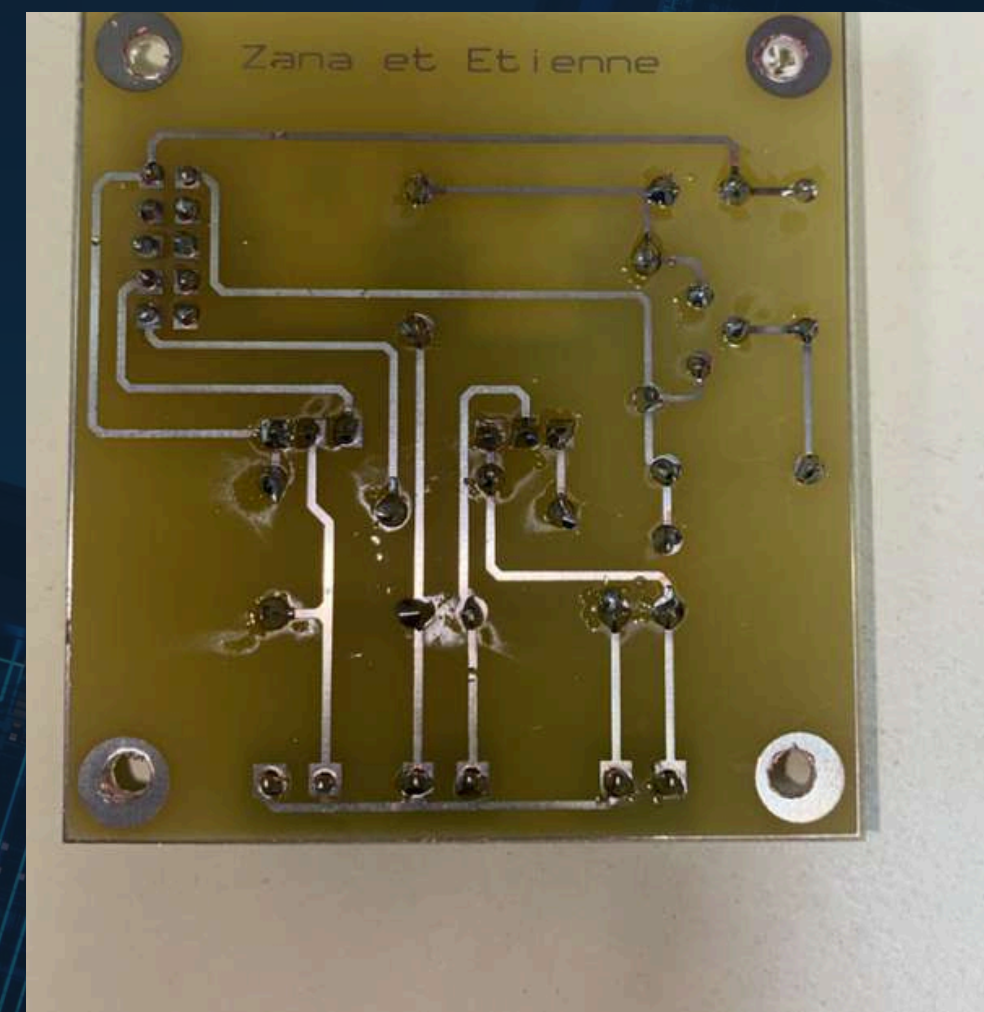
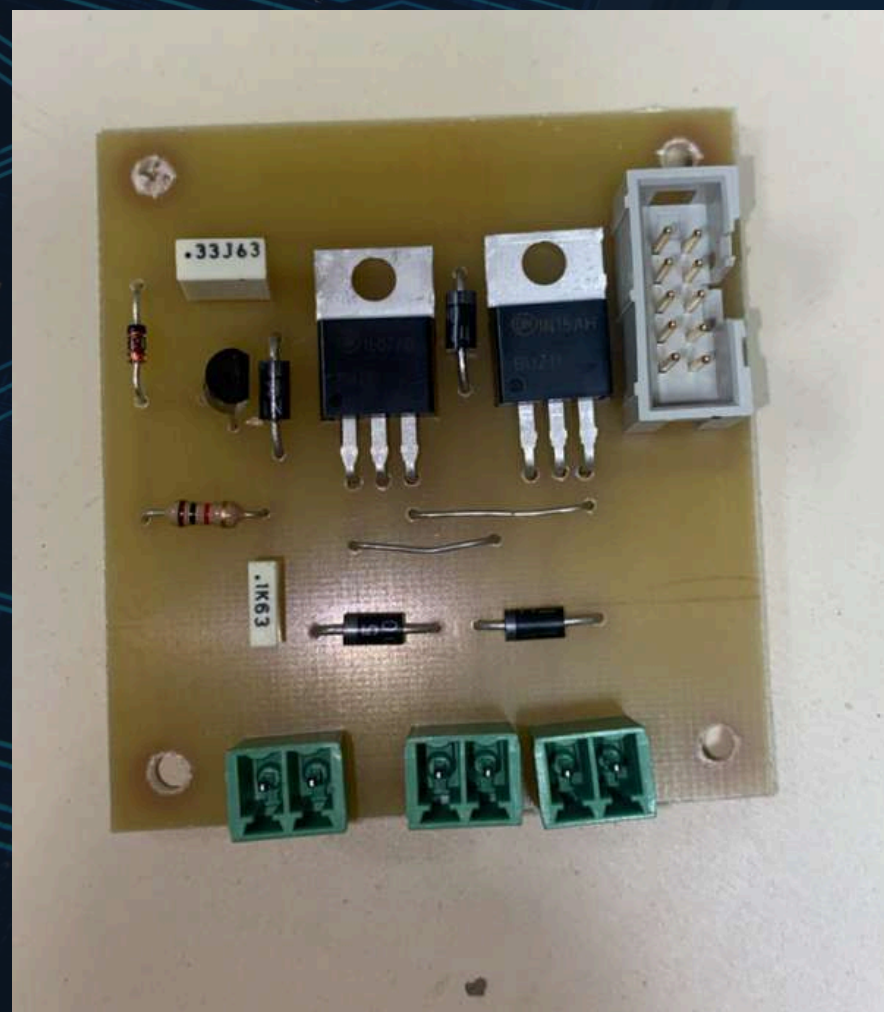


# Typon / routage du circuit imprimé



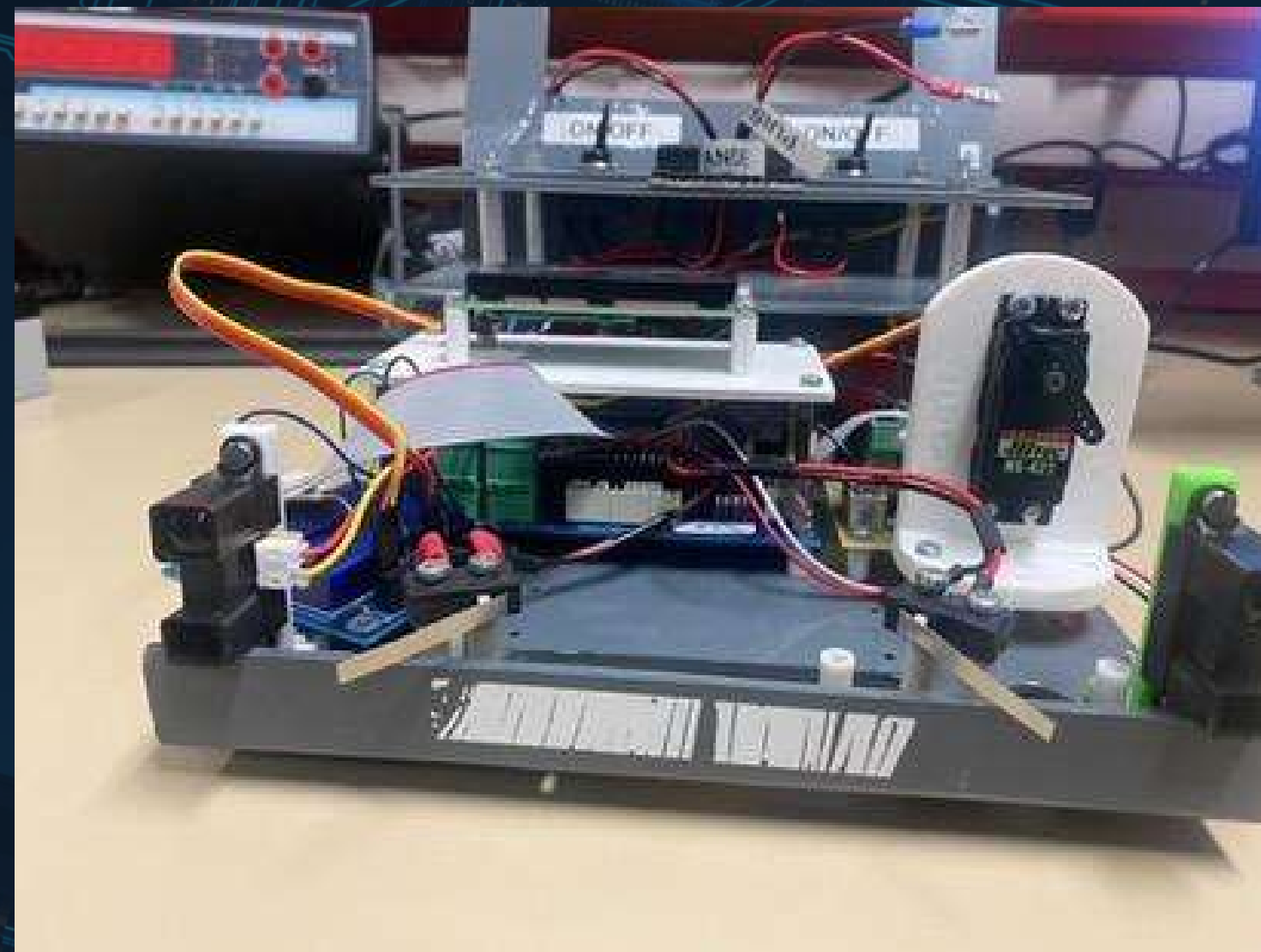


# Fabrication





## Vérification de l'état du robot





## Mise en place des éléments







## Programmes de base

- Conversion d'une tension : fonction map
- Génération d'un signal MLI

```
#include <Mapf.h> //bibliothèque Mapf pour les fonctions de mappage personnalisé
float pot; // Variable pour stocker la valeur lue du potentiomètre
void setup() {
  Serial.begin(9600); //communication série à 9600 bauds pour le débogage.
}

void loop() {
  // Lecture de la valeur analogique du potentiomètre (de 0 à 1023)
  pot = analogRead(A0);

  // Conversion de la valeur analogique en tension (de 0.00 à 5.00 volts) avec mapf
  float volt = mapf(pot, 0.00, 1023.00, 0.00, 5.00);

  // Affichage de la tension sur le moniteur série
  Serial.println(volt);
}
```

```
int mli=5;//Déclaration de la broche pour générer le signal PWM

void setup() {

  pinMode(mli, OUTPUT);
  TCCR0B = TCCR0B & B11111000 | B00000010; // for PWM frequency of 7812.50 Hz
  TCCR2B = TCCR2B & B11111000 | B00000011; // for PWM frequency of 980.39 Hz
}

void loop() {

  analogWrite(mli, 191); //75%
}
```





# Avancer tant que l'espace est libre

```
int bp = 4;           // Bouton start
int mG = 6;           // Moteur gauche
int mD = 5;           // Moteur droit
int moustache = 3;    // Capteur moustache
int valBP;            // État du bouton start
int valSTOP;          // État du capteur moustache

void setup() {
  pinMode(bp, INPUT);      // Bouton en entrée
  pinMode(mG, OUTPUT);     // Moteur gauche en sortie
  pinMode(mD, OUTPUT);     // Moteur droit en sortie
  pinMode(moustache, INPUT); // Capteur moustache en entrée
  TCCR0B = TCCR0B & B11111000 | B00000010; // Fréquence PWM à 7812.50 Hz
}

void loop() {
  valBP = digitalRead(bp); // Lire l'état du bouton start
  valSTOP = digitalRead(moustache); // Lire l'état de la moustache

  if (valBP == 1 && valSTOP == 0) {
    // Si bouton appuyé et espace libre
    analogWrite(mG, 128); // Moteur gauche à 50%
    analogWrite(mD, 128); // Moteur droit à 50%
  }

  if (valSTOP == 1) {
    // Si obstacle détecté
    analogWrite(mG, 0); // Arrêt du moteur gauche
    analogWrite(mD, 0); // Arrêt du moteur droit
  }
}
```





## Déplacement dans un espace avec une bordure

Utilisation de deux capteurs GP2

```
int bp=4;
int mG=6;
int mD=5;
int moustache=3;
int valBP;
int valSTOP;
int capteurG,capteurD;
void setup() {

  pinMode(bp,INPUT); //pinmode sert a définir si c'est une entrée ou une sortie
  pinMode(mG,OUTPUT);
  pinMode(mD,OUTPUT);
  pinMode(moustache,INPUT);
  pinMode(A7,INPUT);
  pinMode(A6,INPUT);
  TCCR0B = TCCR0B & B11111000 | B00000010; // for PWM frequency of 7812.50 Hz
  Serial.begin(9600);
  while(digitalRead(bp)==0);
}

void loop() {
  capteurG=(100-(analogRead(A7)/6));
  capteurD=(100-(analogRead(A6)/6));
  Serial.print("La distance capteur gauche est : "); // affiche le message entre guillemets sur le moniteur série
  Serial.print(capteurG); // affiche la valeur de la variable distance sur le moniteur série
  Serial.print("La distance capteur droite est : "); // affiche le message entre guillemets sur le moniteur série
  Serial.print(capteurD); // affiche la valeur de la variable distance sur le moniteur série
  Serial.println(" cm"); // affiche le message entre guillemets sur le moniteur série
  delay(1000);
  valBP = digitalRead(bp);
  valSTOP = digitalRead(moustache);
  // Avancer à 50 % tant qu'il n'y a pas d'obstacle.
  if (capteurG >= 30 && capteurD >= 30) {
    analogWrite(mG, 128); //moteur gauche 50%
    analogWrite(mD, 128); //moteur droit 50%
  }
  // Si obstacle gauche, tourner à droite.
  if (capteurG < 30) {
    analogWrite(mG, 128); //moteur gauche 50%
    analogWrite(mD, 0); //moteur droit arrêt
  }
  // Si obstacle droit, tourner à gauche.
  if (capteurD < 30) {
    analogWrite(mG, 0); //moteur gauche arrêt
    analogWrite(mD, 128); //moteur droit 50%
  }

  // Temporisation si obstacle (gauche ou droit).
  while (capteurG < 30 || capteurD < 30) {
    delay(200);
    capteurG = (100 - (analogRead(A7) / 6));
    capteurD = (100 - (analogRead(A6) / 6));
  }
  if(valSTOP==1){
    analogWrite(mG, 0); //moteur gauche arrêt
    analogWrite(mD, 0); //moteur droit arrêt
  }
}
```





# Déplacement dans un espace avec une bordure

### Option n°1 : conditions d'arrêt

- 20s depuis le bouton start
- contact moustache

```
int bp = 4; // bouton start
int mG = 6; // moteur Gauche
int mD = 5; // moteur Droite
int moustache = 3; // moustache

int valSTOP;
unsigned long startTime; // Variable pour stocker le temps de démarrage

void setup() {
  pinMode(bp, INPUT);
  pinMode(mG, OUTPUT);
  pinMode(mD, OUTPUT);
  pinMode(moustache, INPUT);

  TCCR0B = TCCR0B & B11111000 | B00000010; // Fréquence PWM.

  while (digitalRead(bp) == 0);

  // Enregistrer le temps de démarrage
  startTime = millis();
}

void loop() {
  if (millis() - startTime >= 16000) {
    analogWrite(mG, 0); // Arrêt moteur gauche
    analogWrite(mD, 0); // Arrêt moteur droit
    while (true); // Blocage du programme
  }
  // Vérifier si la moustache détecte un contact
  valSTOP = digitalRead(moustache);
  if (valSTOP == 1) {
    analogWrite(mG, 0); // Arrêt moteur gauche
    analogWrite(mD, 0); // Arrêt moteur droit
    while (true); // Blocage du programme
  }
}
```





# Déplacement dans un espace avec une bordure

Option n°2 : LEDS rouge et verte

```
int bp = 4; // bouton start
int mG = 6; // moteur Gauche
int mD = 5; // moteur droit
int moustache = 3;
int ledVerte = 7;
int ledRouge = 6;
int valSTOP;
int capteurG, capteurD;
void setup() {
  pinMode(bp, INPUT);
  pinMode(mG, OUTPUT);
  pinMode(mD, OUTPUT);
  pinMode(moustache, INPUT);
  pinMode(A7, INPUT);
  pinMode(A6, INPUT);
  pinMode(ledVerte, OUTPUT);
  pinMode(ledRouge, OUTPUT);

  TCCR0B = TCCR0B & B11111000 | B00000010; // Fréquence PWM.
  Serial.begin(9600);
  while (digitalRead(bp) == 0);
}

void loop() {
  // Lire les distances des capteurs
  capteurG = (100 - (analogRead(A7) / 6));
  capteurD = (100 - (analogRead(A6) / 6));

  // Avancer si pas d'obstacles
  if (capteurG >= 30 && capteurD >= 30) {
    analogWrite(mG, 128); // Moteur gauche à 50 %
    analogWrite(mD, 128); // Moteur droit à 50 %
    digitalWrite(ledVerte, HIGH);
    digitalWrite(ledRouge, LOW);
  }
  // Tourner à droite si obstacle gauche
  else if (capteurG < 30) {
    analogWrite(mG, 128);
    analogWrite(mD, 0);
    digitalWrite(ledVerte, LOW);
    digitalWrite(ledRouge, HIGH);
  }
  // Tourner à gauche si obstacle droit
  else if (capteurD < 30) {
    analogWrite(mG, 0);
    analogWrite(mD, 128);
    digitalWrite(ledVerte, LOW);
    digitalWrite(ledRouge, HIGH);
  }
}
```





## Servomoteur

```
#include <Servo.h>
int bp = 4;
int mG = 6;
int mD = 5;
int valBP;
int valSTOP;
int capteurG, capteurD; // déclaration des capteurs
Servo directionServo; // Créer un objet pour le servo
void setup() {
  pinMode(bp, INPUT); // Définir le bouton BP comme entrée
  pinMode(mG, OUTPUT);
  pinMode(mD, OUTPUT);
  pinMode(moustache, INPUT);
  pinMode(A7, INPUT);
  pinMode(A6, INPUT);

  TCCR0B = TCCR0B & B11111000 | B00000010; // Pour fréquence PWM de 7812.50 Hz
  Serial.begin(9600);
  // Initialiser le servo
  directionServo.attach(9);
  directionServo.write(90); // Position initiale (tout droit)
  delay(1000);

  // Attendre que l'utilisateur appuie sur le bouton BP pour commencer
  while (digitalRead(bp) == 0) {
    delay(100);
  }
}

void loop() {
  capteurG = (100 - (analogRead(A7) / 6));
  capteurD = (100 - (analogRead(A6) / 6));

  // Si aucun obstacle, avancer tout droit
  if (capteurG >= 30 && capteurD >= 30) {
    analogWrite(mG, 128);
    analogWrite(mD, 128);
    directionServo.write(90); // Avancer tout droit
  }
  // Si l'obstacle est à gauche, tourner à droite
  if (capteurG < 30) {
    analogWrite(mG, 0);
    analogWrite(mD, 128);
    directionServo.write(180); // Tourner à droite
    lcd.clear();
    lcd.print("Tourner à droite");
  }
  // Si l'obstacle est à droite, tourner à gauche
  if (capteurD < 30) {
    analogWrite(mG, 128);
    analogWrite(mD, 0);
    directionServo.write(0); // Tourner à gauche
  }
}
```





## Affichage LCD

```
#include <Servo.h>
#include <Wire.h> //Communication I2C
#include <LiquidCrystal_I2C.h>

int bp = 4;
int mG = 6;
int mD = 5;
int moustache = 3;
int valBP;
int valSTOP;

int capteurG, capteurD; // déclaration des capteurs
int distance;
unsigned long startTime; // Variable pour stocker le temps de démarrage
Servo directionServo; // Créer un objet pour le servo
LiquidCrystal_I2C lcd(0x27, 16, 2);

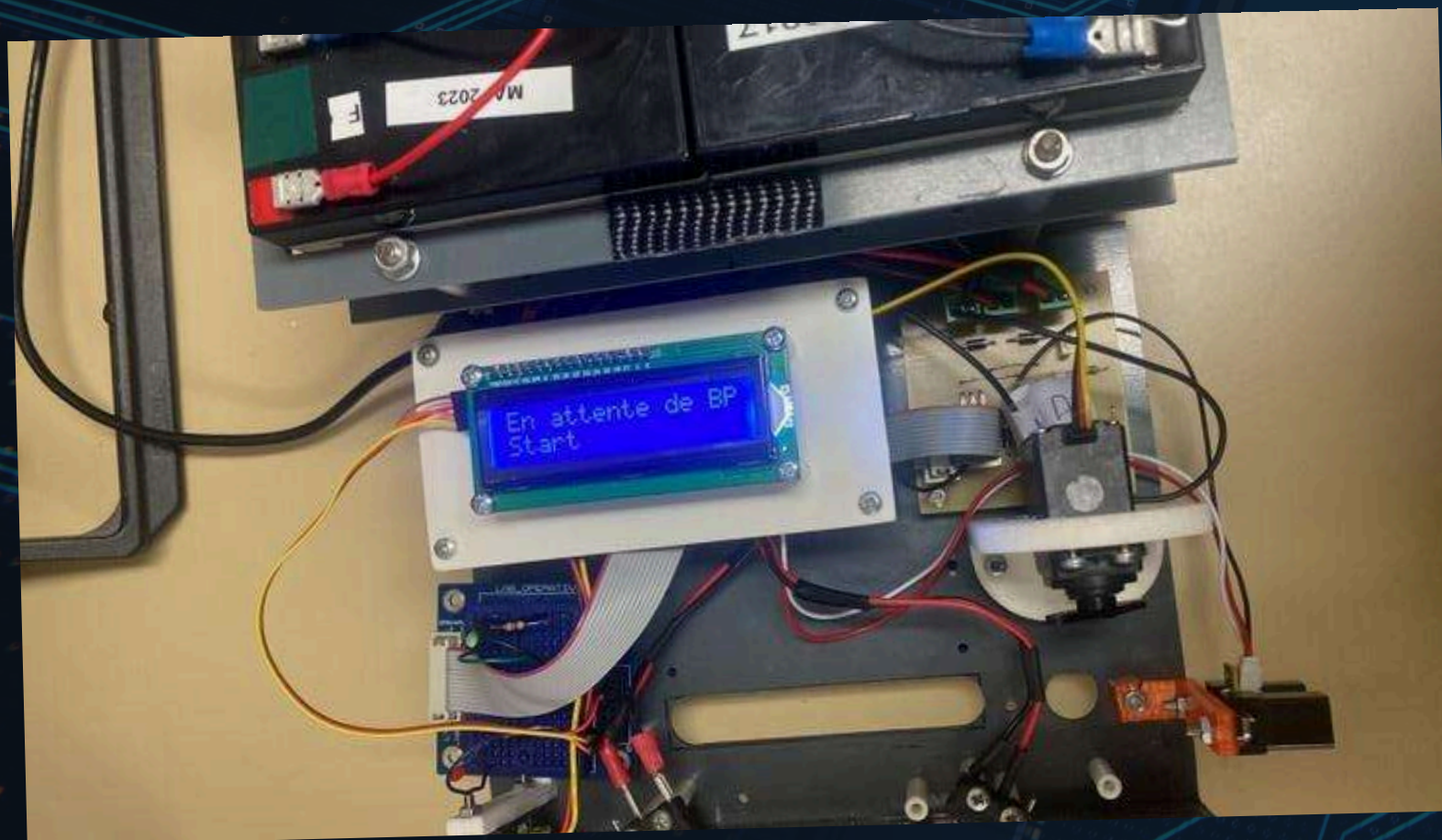
void loop() {
  capteurG = (100 - (analogRead(A7) / 6));
  capteurD = (100 - (analogRead(A6) / 6));
  valBP = digitalRead(bp);
  valSTOP = digitalRead(moustache);

  // Si aucun obstacle, avancer tout droit
  if (capteurG >= 30 && capteurD >= 30) {
    analogWrite(mG, 128);
    analogWrite(mD, 128);
    directionServo.write(90); // Avancer tout droit
    lcd.clear();
    lcd.print("Tout droit");
  }
  // Si l'obstacle est à gauche, tourner à droite
  else if (capteurG < 30) {
    analogWrite(mG, 0);
    analogWrite(mD, 128);
    directionServo.write(180); // Tourner à droite
    lcd.clear();
    lcd.print("Tourner a droite");
  }
  // Si l'obstacle est à droite, tourner à gauche
  else if (capteurD < 30) {
    analogWrite(mG, 128);
    analogWrite(mD, 0);
    directionServo.write(0); // Tourner à gauche
    lcd.clear();
    lcd.print("Tourner a gauche");
  }
}

// Vérification du temps écoulé
if (millis() - startTime >= 160000) {
  analogWrite(mG, 0); // Arrêt moteur gauche
  analogWrite(mD, 0); // Arrêt moteur droit
  lcd.clear();
  lcd.print("Temps ecoulé");
  while (true); // Arrêt complet après le temps écoulé
}
```



# Affichage LCD



```
void setup() {  
  pinMode(bp, INPUT); // Définir le bouton BP comme entrée  
  pinMode(mG, OUTPUT);  
  pinMode(mD, OUTPUT);  
  pinMode(moustache, INPUT);  
  pinMode(A7, INPUT);  
  pinMode(A6, INPUT);  
  
  TCCR0B = TCCR0B & B11111000 | B00000010; // Pour fréquence PWM de 7812.50 Hz  
  Serial.begin(9600);  
  
  // Initialiser le servo  
  directionServo.attach(9);  
  directionServo.write(90); // Position initiale (tout droit)  
  delay(1000);  
  
  // Initialisation de l'écran LCD  
  lcd.init();  
  lcd.backlight();  
  lcd.setCursor(0, 0);  
  lcd.print("En attente de BP Start");  
  
  // Attendre que l'utilisateur appuie sur le bouton BP pour commencer  
  while (digitalRead(bp) == 0) {  
    delay(100);  
  }  
  
  // Afficher "Démarrage" une fois que le BP est appuyé  
  lcd.clear();  
  lcd.print("Demarrage...");  
  delay(1000);  
}
```



## Suivre une paroi à 40 cm

```
int bp = 4;
int mG = 6;
int mD = 5;
int valBP;
int valSTOP;
int E1;
int vd;
int capteurG, capteurD; // déclaration des capteurs

void setup() {
  pinMode(bp, INPUT); // Définir le bouton BP comme entrée
  pinMode(mG, OUTPUT);
  pinMode(mD, OUTPUT);
  pinMode(A7, INPUT);
  pinMode(A6, INPUT);

  TCCR0B = TCCR0B & B11111000 | B00000010; // Pour fréquence PWM de 7812.50 Hz
  Serial.begin(9600);
  // Attendre que l'utilisateur appuie sur le bouton BP pour commencer
  while (digitalRead(bp) == 0) {
    delay(100);
  }
}

void loop() {
  capteurD = (100 - (analogRead(A7) / 6));
  capteurG = (100 - (analogRead(A6) / 6));
  valBP = digitalRead(bp);
  valSTOP = digitalRead(moustache);
  E1=capteurD-40;
  Serial.print("valeur E1:");
  Serial.println(E1);
  vd=128+5*E1;
  Serial.print(" ");
  Serial.println(vd);

  if(vd>255){
    vd=255;
  }
  if(vd<0){
    vd=0;
  }
  analogWrite(mG,128 );
  analogWrite(mD, vd);
}
```





FIN

Zana DIAMOUTENE & Etienne RUAULT