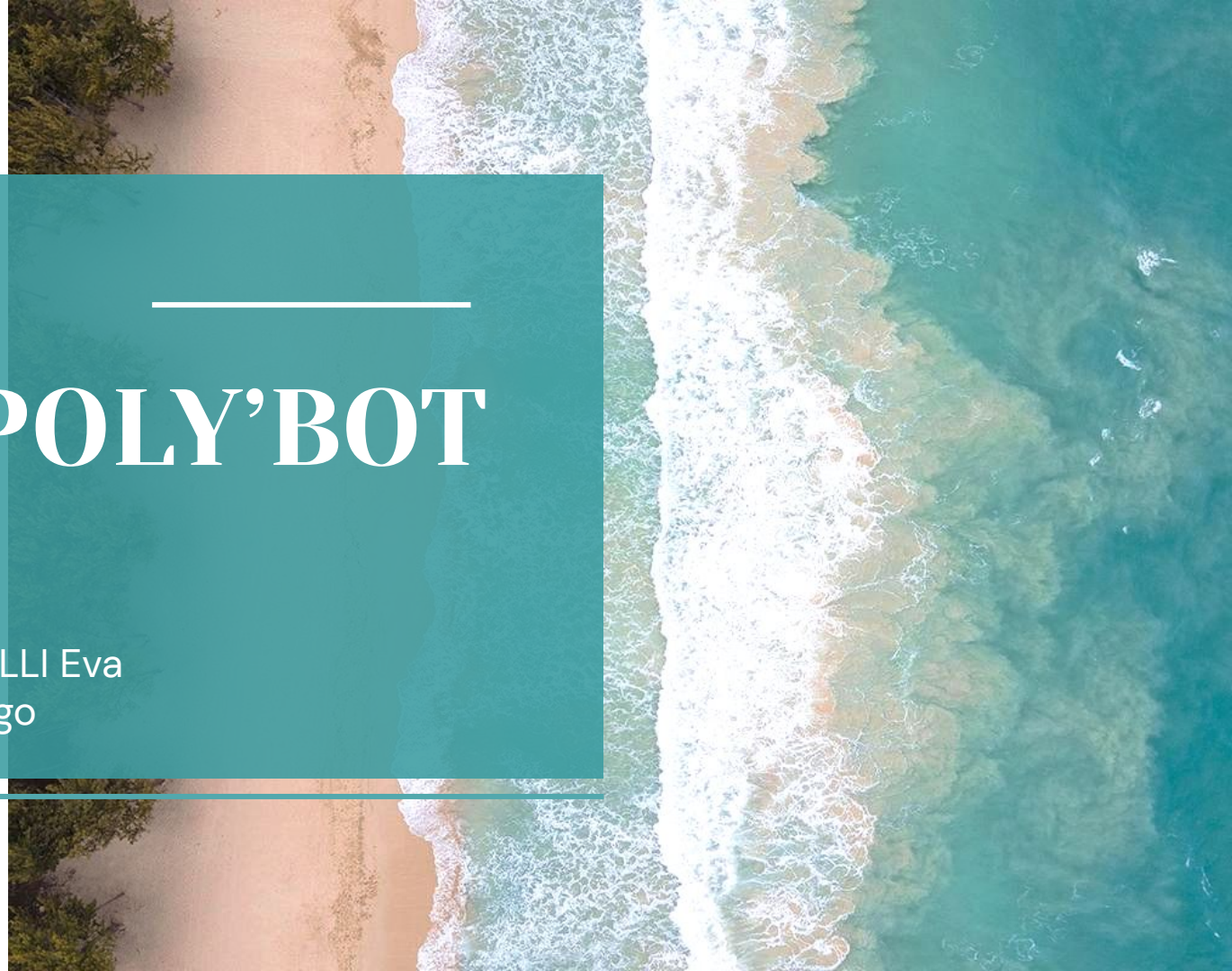
POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

# BEACH POLY'BOT

LABAUVIE--RAFFAELLI Eva
DENIS--MARTIN Hugo



# SOMMAIRE

POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

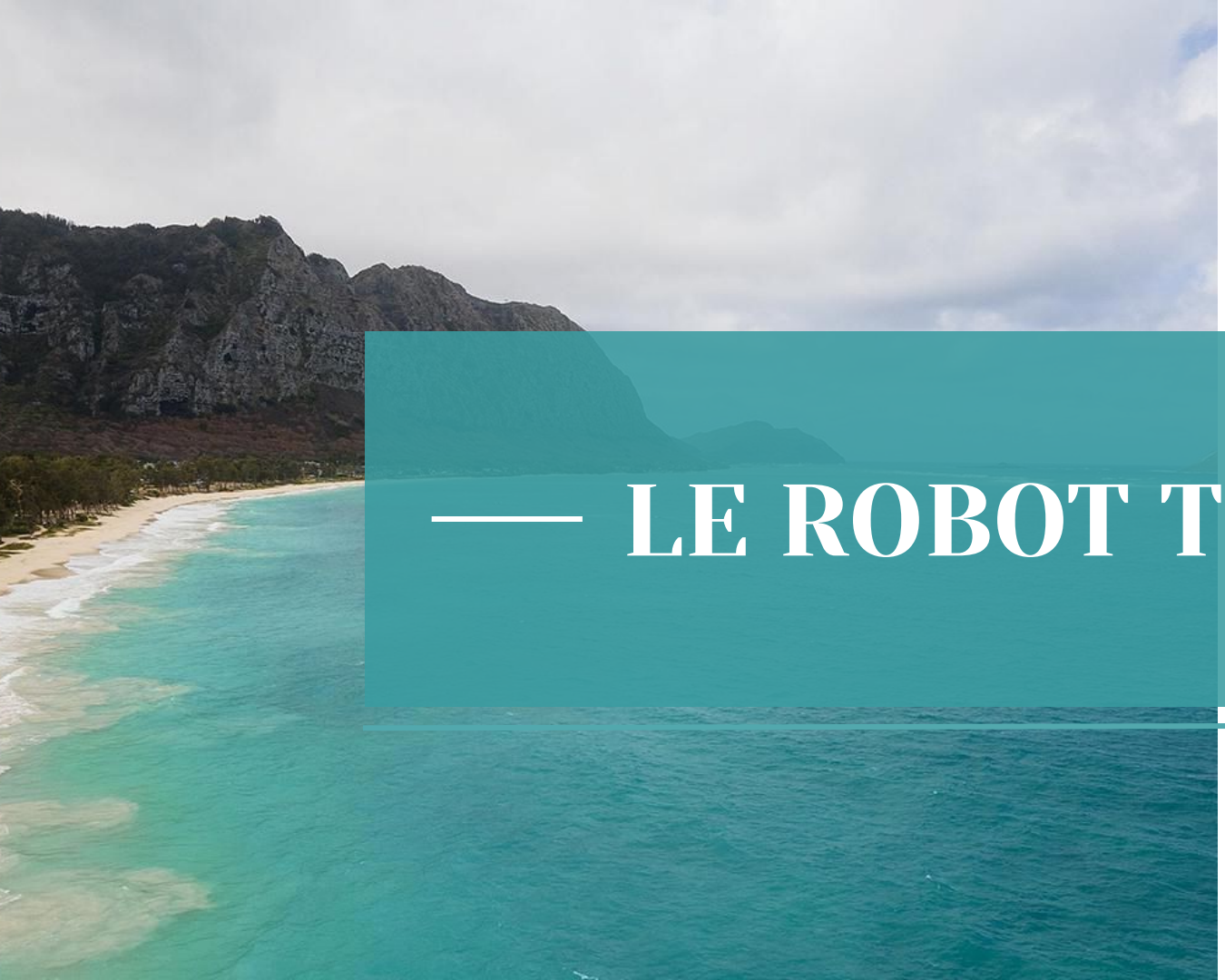
# 01

# INTRODUCTION

# BEACH POLY'BOT

*Robot tracteur*

*Dispositif de collecte*



POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

# 02

# LE ROBOT TRACTEUR

# ROBOT TRACTEUR

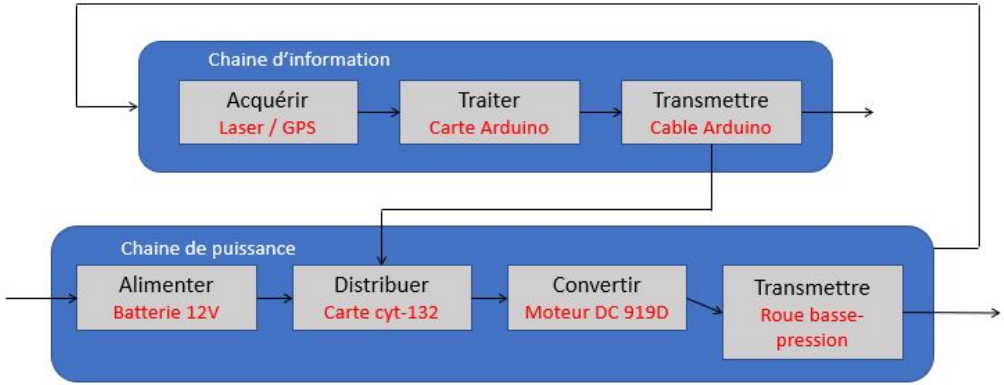
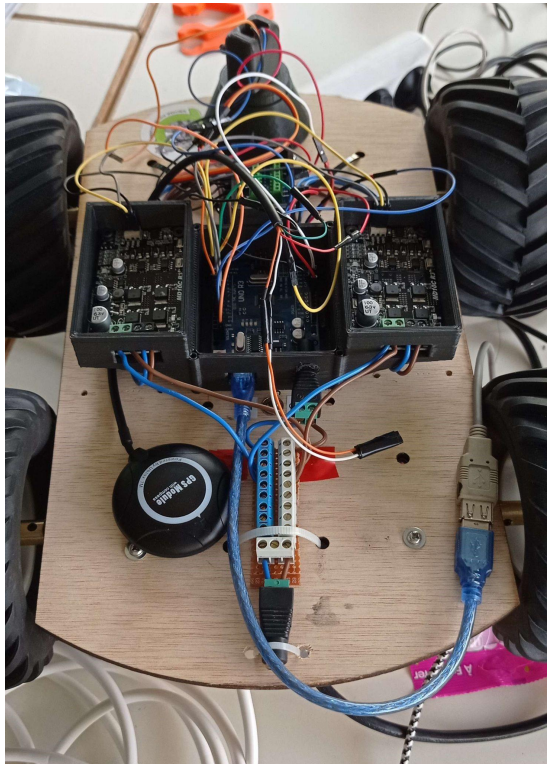
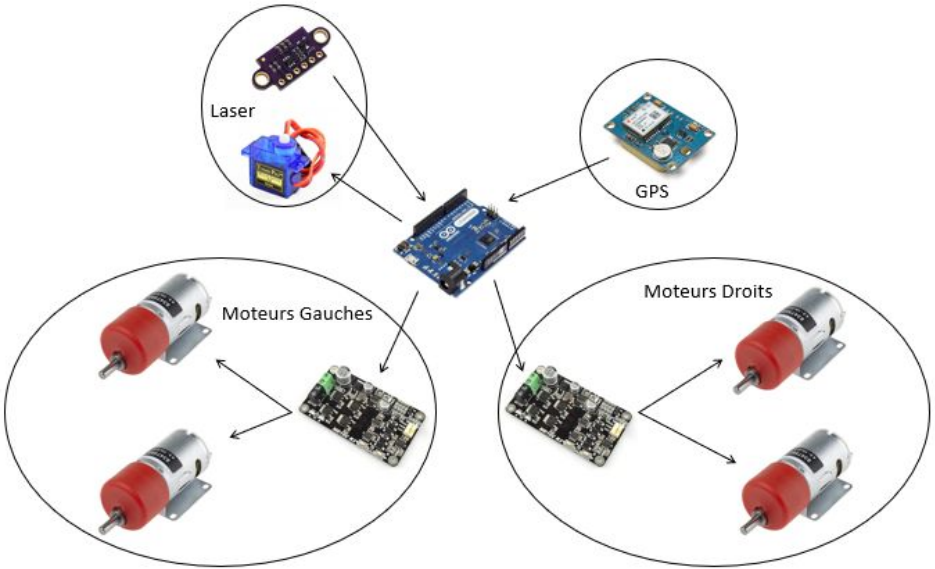



Schéma Fonctionnel



# ROBOT TRACTEUR




Partie Commande:

- Carte de commande Arduino

Partie Laser:

- Laser VL53L0
- Servomoteur SG90

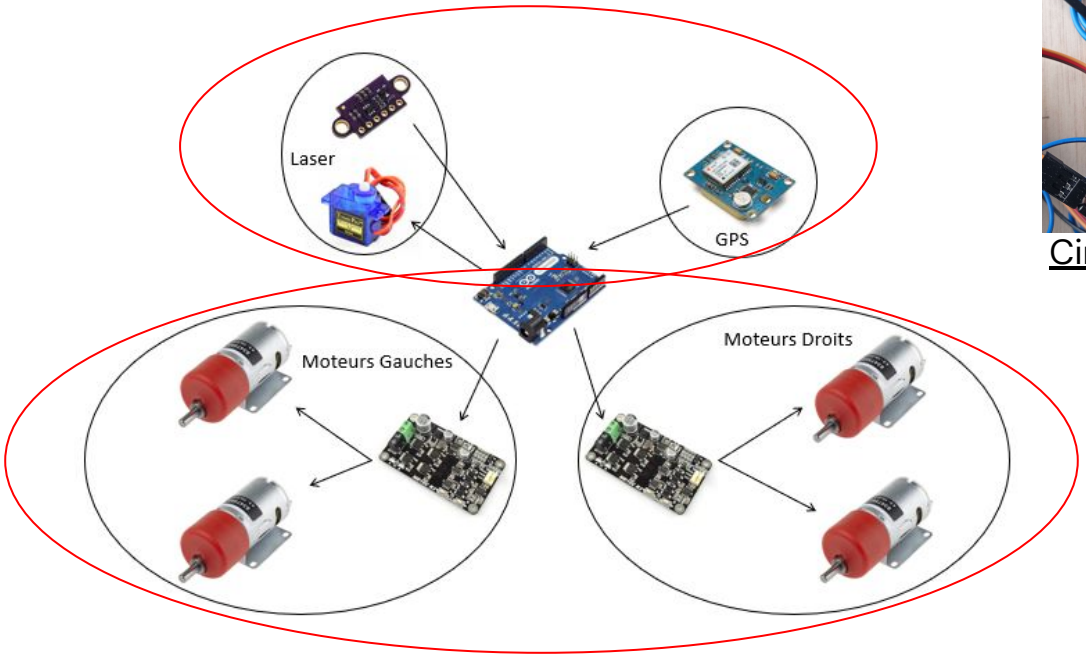
Partie Positionnement:

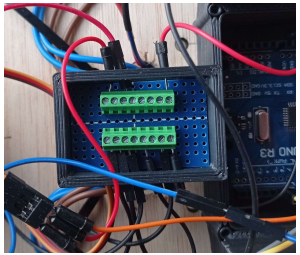
- GPS Ublox 7M

Partie Puissance:

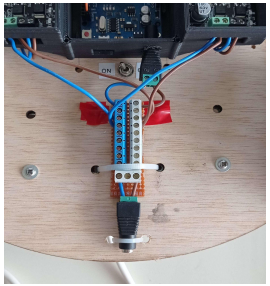
- Carte de puissance cyt-132 (x2)
- Moteurs 919D (x4)

# ROBOT TRACTEUR



Circuit Commande 5V



Circuit Puissance 12V

POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

# 03

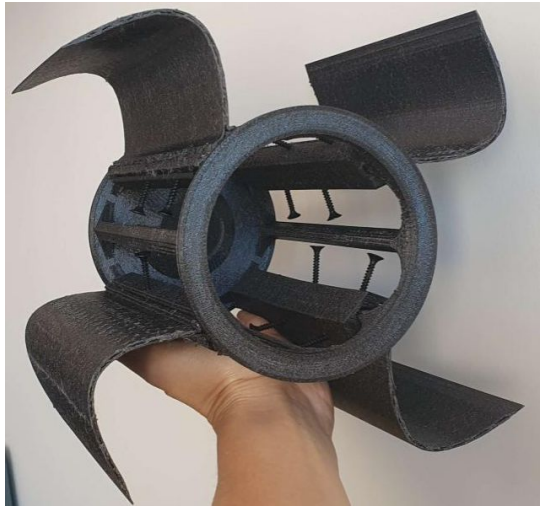
# DISPOSITIF DE COLLECTE

# DISPOSITIF DE COLLECTE

- Placée à l'arrière du robot
- Dans un premier temps pour collecter les déchets et les stocker
- *Dans un second temps, pour les trier*

# ROUE DE COLLECTE

- Récolter le sable et les déchets s'y trouvant
- Les envoyer sur un convoyeur par le biais d'une pente



# TAPIS DE TRANSPORT

- Récupère les déchets rejetés par la roue
- Dans un premier temps, pour envoyer les déchets dans un bac
- *Dans un second temps, pour permettre de les trier*

POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

# 04

# DEMONSTRATION

POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

# 05

# LES AMELIORATIONS

# ROBOT TRACTEUR

***Résultats actuels*** :

- Scan avec peu de points
- Evitement opérationnel
- Gps en cours

***Améliorations*** :

- Structure hermétique
- Moteurs adapté (couple)
- Autres schémas de déplacements
- Système de raccord avec le ramassage

# DISPOSITIF DE COLLECTE

***Résultats actuels*** :

- Forme générale et solution trouvées
- Encore à l'état de prototype
- Difficultés à mettre la carte en fonctionnement : manque de reconnaissance d'image

***Améliorations*** :

- Refaire les pièces de façons optimales pour permettre une meilleure unité du dispositif
- Intégrer le système de tri
- Intégrer la reconnaissance d'image

POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

# 06

# CONCLUSION

# REMERCIEMENTS

*Nous remercions* :

Pascal MASSON

Christian PETERS

Xavier LEBRETON

Sébastien ROTHUT

Axel FAUVEL

Frédéric JUAN

*Pour leur aide durant la réalisation de notre projet tout au long de l'année.*

POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

# MERCI POUR VOTRE ATTENTION

POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

# ANNEXES

# Code Main()

```
#include "Capteurdistance.hpp"
#include "Adafruit_VL53L0X.h"
#include "Servo.h"
#include "Gps.hpp"
#include "TinyGPS++.h"
#include "SoftwareSerial.h"
#include "Mouvements.hpp"

//Servo
Servo servoinf;
//Moteurs Droit
int PWDD = 5;
int DIRD = 12;
//Moteurs Gauche
int PWDG = 6;
int DIRG = 13;
//Interupteur
int switchgps = 7;
//Library
Mouvements mvt = Mouvements();
Capteurdistance cd = Capteurdistance();
Gps gps = Gps();

unsigned long startTime = 0;
int positionMin = 0;
const int DEPLACEMENT_SERVO = 100;
float Lat = 0;
float Lon = 0;

static const int RXPin = 2, TXPin = 3;
static const uint32_t GPSBaud = 9600;
// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);
```

```
void setup() {
  Serial.begin(9600);
  Serial.println(F("Setup Started"));

  //Motors Setup
  pinMode(PWDD, OUTPUT);
  pinMode(DIRD, OUTPUT);
  pinMode(PWDG, OUTPUT);
  pinMode(DIRG, OUTPUT);
  //Interupter Setup
  pinMode(switchgps, INPUT_PULLUP);
  //Arm servo-motors Setup
  servoinf.attach(9);
  servoinf.write(65);
  delay(100);
  //Serial Port Setup
  while (!Serial) { delay(1); }  //This part does a test on th
  //Laser Captor VL53L0X Setup
  if (!cd.begin()) {
    Serial.println(F("Failed to boot Captor VL53L0X"));
    while (1)
      ;
  }
  int distancelaser;
  Serial.println(F("Succeeded to boot Captor VL530X"));
  Serial.println(F("Setup Finished"));
}
```

```
void loop() {

  mvt.forward();

  if (digitalRead(switchgps) == HIGH) {
    Lat = gps.latitude();
    Lon = gps.longitude();
    Serial.print("Latitude: ");Serial.println(Lat);
    Serial.print("Longitude: ");Serial.println(Lon);
  }


  if (millis() - startTime > DEPLACEMENT_SERVO) {
    cd.continuousScan(servoinf);
    cd.angleIncrement();
    int distancelaser = cd.getDist();

    if (distancelaser < 200) {
      Serial.println(F("Object in range"));
      mvt.avanceBackward(1000);
      cd.scanSweep(servoinf);
      positionMin = cd.getMin();

      if (positionMin <= 4) {
        mvt.avanceLeft(1000);
      }

      else {
        mvt.avanceRight(1000);
      }

      cd.setAngle(4);
    }

    startTime = millis();
  }

}
```

# Code Mouvements

```cpp
#include "Mouvements.hpp"

Mouvements::Mouvements(){};

/*
For the PWD, 255 means 98 rpm
so 128 is for 49 rpm
By activating or not DIR, we can change the direction of rotation.
*/

void Mouvements::off() {
  analogWrite(PWDD, 0);
  digitalWrite(DIRD, HIGH);
  analogWrite(PWDG, 0);
  digitalWrite(DIRG, HIGH);
}

void Mouvements::forward() {
  analogWrite(PWDD, 32);
  digitalWrite(DIRD, HIGH);
  analogWrite(PWDG, 32);
  digitalWrite(DIRG, LOW);
}

void Mouvements::backward() {···

void Mouvements::left() {···

void Mouvements::right() {···

void Mouvements::avanceForward(int temps) {···

void Mouvements::avanceBackward(int temps) {···

void Mouvements::avanceLeft(int temps) {···

void Mouvements::avanceRight(int temps) {···
```

```cpp
#ifndef MOUVEMENTS_H
#define MOUVEMENTS_H
#include "Arduino.h"

/*

*/

class Mouvements {
public:
  Mouvements();
  unsigned long InstantTime;  //unsigned long permet d'avoir des chiffre allant jusqu'à 2^32 - 1
  const int PWDD = 5;
  const int DIRD = 12;
  const int PWDG = 6;
  const int DIRG = 13;
  void off();
  void forward();
  void backward();
  void left();
  void right();
  void avanceForward(int temps);
  void avanceBackward(int temps);
  void avanceRight(int temps);
  void avanceLeft(int temps);
};

#endif
```

# Code Laser

```
int Capteurdistance::distance() {
  VL53L0X_RangingMeasurementData_t measure;
  rangingTest(&measure, false);
  return measure.RangeMilliMeter;
  Serial.println("Mesure distance");
  Serial.println(measure.RangeMilliMeter);
}
```

```
int Capteurdistance::scanSweep(Servo servoinf) {
  Serial.println(F("==> scanSweep"));
  int i = 0;
  min_v = 9000;
  min_i = 0;

  for (int servoInfPosition = infAngleMin; servoInfPosition <= infAngleMax; servoInfPosition += infPas) {
    servoinf.write(servoInfPosition);
    delay(150);
    scanTableau[i] = distance();
    Serial.print(F("La valeur de i = "));
    Serial.print(i);
    Serial.print(F(" est : "));
    Serial.println(scanTableau[i]);
    if (scanTableau[i] < min_v) {
      min_v = scanTableau[i];
      min_i = i;
    }
    i++;
  }

  Serial.print("La valeur min est ");
  Serial.print(min_v);
  Serial.print(" et elle est en position ");
  Serial.println(min_i);
  Serial.println(F("==> scanSweep finished"));
  servoinf.write(65);
}
```

```
int Capteurdistance::continuousScan(Servo servoinf) {
  int k = 0;

  while (k <= anglePosition) {
    servoInfPosition = infAngleMin + k * infPas;
    k++;
  }

  servoinf.write(servoInfPosition);
  dist = distance();
  return (dist);
}

int Capteurdistance::angleIncrement() {

  if (sensScan == true) {
    anglePosition++;
    if (anglePosition >= lenghtScanTableau + 1) {
      sensScan = false;
      anglePosition--;
    }
  }

  else if (sensScan == false) {
    anglePosition=anglePosition - 1;
    if (anglePosition < 0) {
      sensScan = true;
      anglePosition++;
    }
  }
}
```

```
#ifndef CAPTEURDISTANCE_H
#define CAPTEURDISTANCE_H
#include "Arduino.h"
#include "Adafruit_VL53L0X.h"
#include "Servo.h"

/*

*/

class Capteurdistance : public Adafruit_VL53L0X {
public:
  Capteurdistance();
  int distance();
  int getAngle();
  void setAngle(int angle);
  int getDist();
  int getMin();
  int scanSweep(Servo servoinf);
  int continuousScan(Servo servoinf);
  int angleIncrement();
  int servoInfPosition = 0;
  int servoSupPosition = 0;
  int dist;
  static const int lenghtScanTableau = 9;
  int anglePosition = 0;
  int infAngleMin = 1;
  int infAngleMax = 129;
  int infPas = 16;
  int supAngleMin = 80;
  int supAngleMax = 120;
  int supPas = 10;
  int scanTableau[lenghtScanTableau];
  int min_v = 9000;
  int min_i = 0;
  bool sensScan = true;
};

#endif
```

# Code GPS

```cpp
#include "Gps.hpp"

Gps::Gps() : ss(RXPin, TXPin) {}

void Gps::loop() {
  printFloat(gps.location.lat(), gps.location.isValid(), 11, 6);
  printFloat(gps.location.lng(), gps.location.isValid(), 12, 6);
  Serial.println();
  smartDelay(1000);
  if (millis() > 5000 && gps.charsProcessed() < 10)
    Serial.println(F("No GPS data received: check wiring"));
}

float Gps::latitude() {
  return gps.location.lat();
}

float Gps::longitude() {
  return gps.location.lng();
}

void Gps::smartDelay(unsigned long ms) {
  unsigned long start = millis();
  do {
    while (ss.available())
      gps.encode(ss.read());
  } while (millis() - start < ms);
}

void Gps::printFloat(float val, bool valid, int len, int prec) {···
```

```cpp
void Gps::printFloat(float val, bool valid, int len, int prec) {
  if (!valid) {
    while (len-- > 1)
      Serial.print('*');
    Serial.print(' ');
  } else {
    Serial.print(val, prec);
    int vi = abs((int)val);
    int flen = prec + (val < 0.0 ? 2 : 1);  // . and -
    flen += vi >= 1000 ? 4 : vi >= 100 ? 3
                           : vi >= 10  ? 2
                                       : 1;
    for (int i = flen; i < len; ++i)
      Serial.print(' ');
  }
  smartDelay(0);
}
```

```cpp
#ifndef GPS_H
#define GPS_H

#include "TinyGPS++.h"
#include "SoftwareSerial.h"

class Gps {
  public:
    Gps();
    void loop();
    float latitude();
    float longitude();

  private:
    static const int RXPin = 2, TXPin = 3;
    static const uint32_t GPSBaud = 9600;
    TinyGPSPlus gps;
    SoftwareSerial ss;
    void smartDelay(unsigned long ms);
    void printFloat(float val, bool valid, int len, int prec);
};

#endif
```