TEKBOT ROBOTICS CHALLENGE 2025

Test 1 : Électronique, Informatique et Mécanique

TEAM IMSP

PLAN GÉNÉRAL

Introduction

- ✓ Concevoir un robot qui collecte et trie les déchets.
- ✓ Développer des compétences essentielles à l'issue de chaque test.
- ✓ Former des équipes capables de proposer une solution durable et technologique pour l’Afrique.

02. ELECTRONIQUE

Test input : Gyroscope et accéléromètre

PLAN


Présentation &
Fonctionnalités
02
Résultats
01
Montage &
Implémentation
03

Présentation & Fonctionnalités

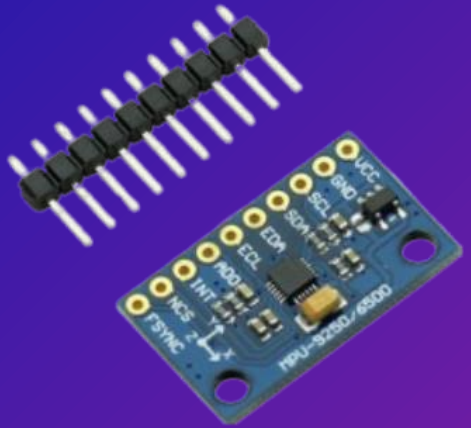
Présentation

- Étude de l’orientation d’un objet dans l’espace
- Utilisation d’un capteur gyroscope + accéléromètre
- Affichage des données sur écran LCD via Arduino

Présentation & Fonctionnalités

Fonctionnement du capteur MPU6050

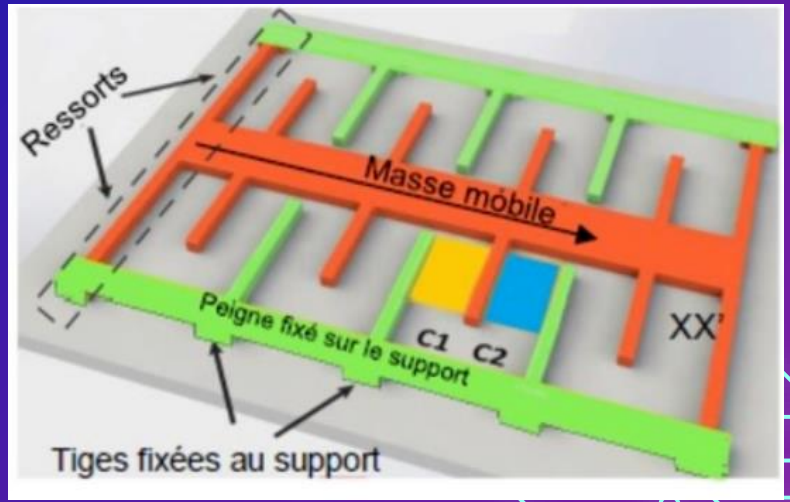
- 3 axes d’accélération linéaire
- 3 axes de vitesse angulaire (gyroscope)
- Communication via I2C



Présentation & Fonctionnalités

L’accéléromètre

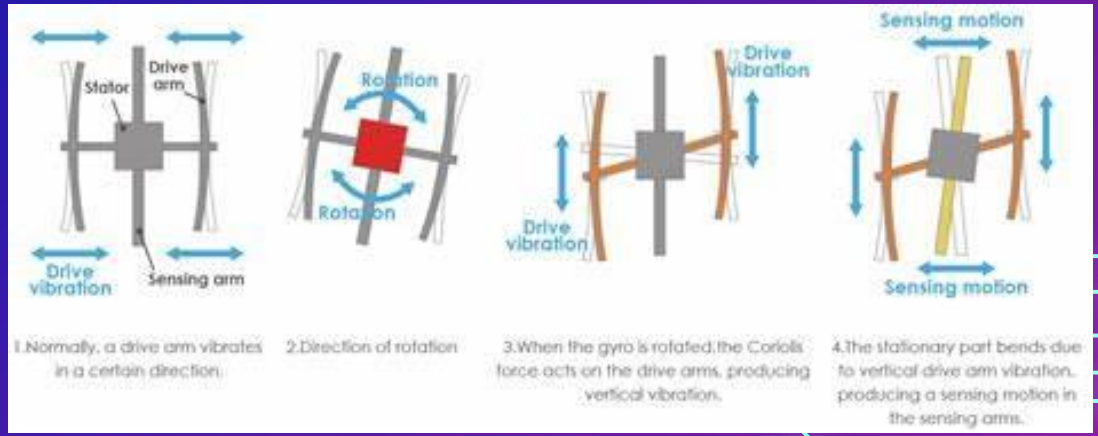
- Masse sismique suspendue
- Variation de capacité : acceleration
- Axes X, Y, Z mesurés indépendamment



Présentation & Fonctionnalités

Le gyroscope

- Utilise l'effet Coriolis
- Détection de vitesse angulaire
- Axes X, Y, Z indépendamment



Présentation & Fonctionnalités

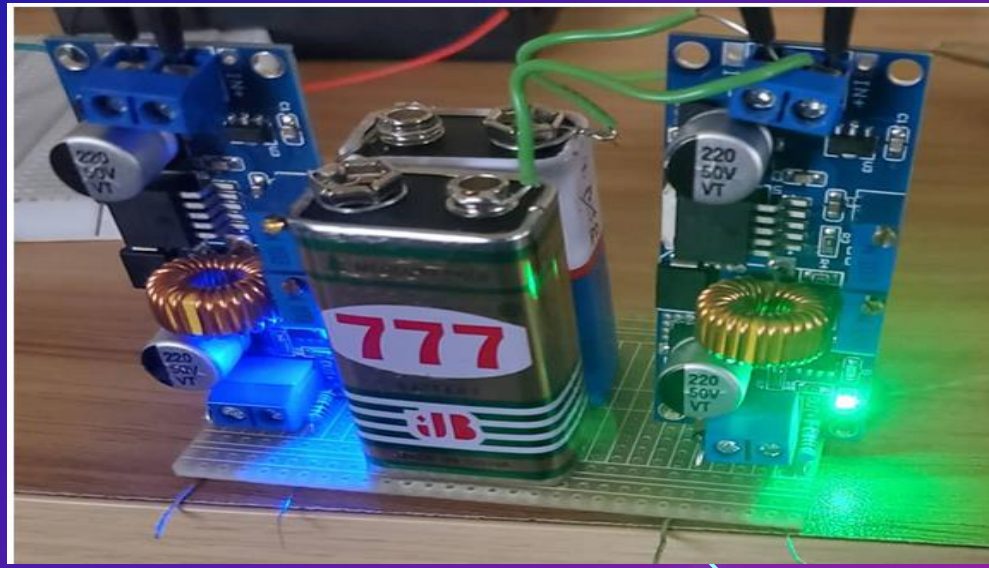
Architecture matérielle

- Capteur MPU6050
- Détection de vitesse angulaire
- Écran LCD 20x4 + module I2C
- Alimentation externe stabilisée

MONTAGE & IMPLÉMENTATION

Conception de l’alimentation

- 2 piles de 9V en série
- Régulateur buck XL4015
- Diode Zener + résistance de sécurité

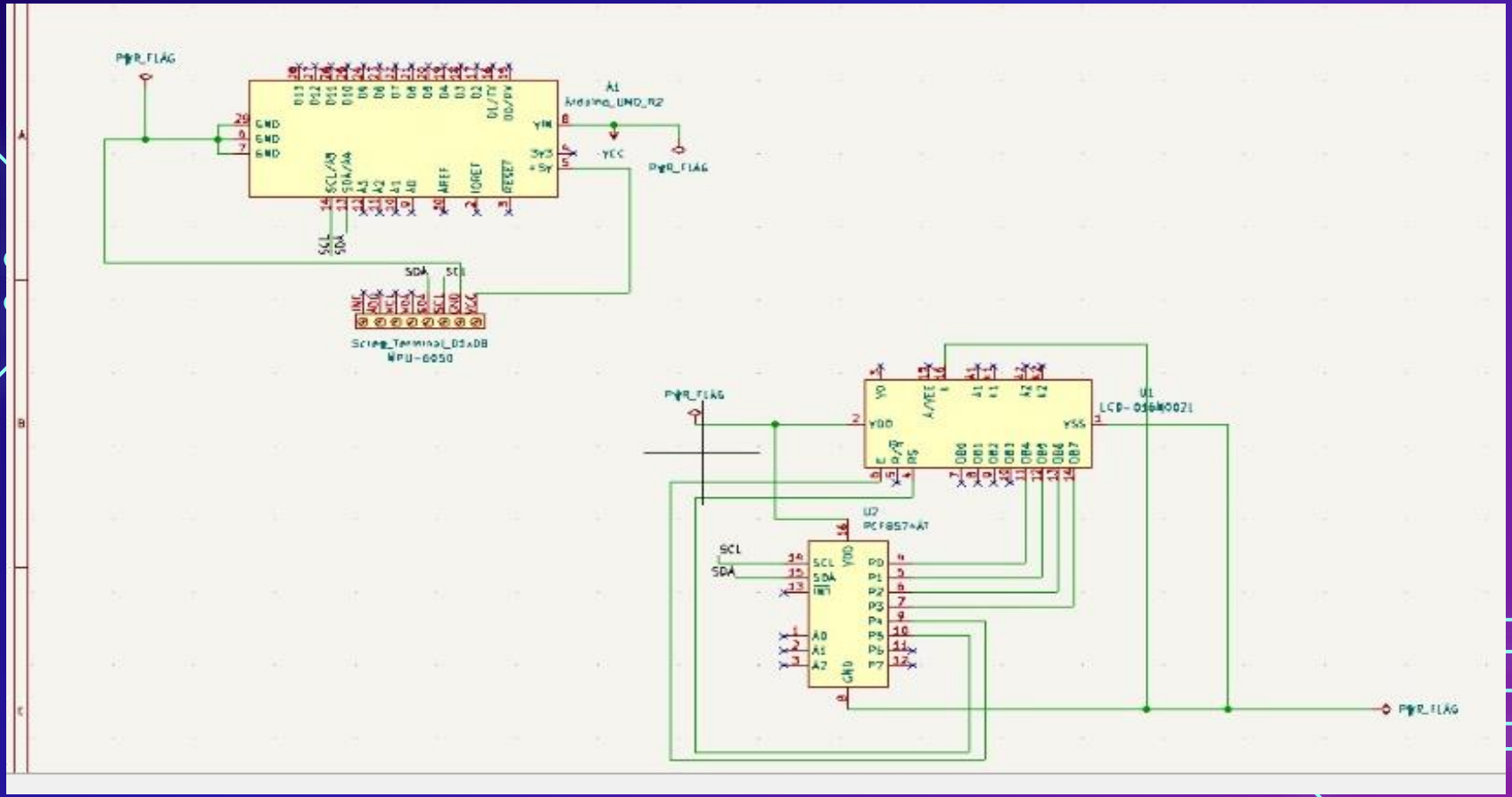


MONTAGE & IMPLÉMENTATION

Schéma de câblage

- LCD avec module I2C : SDA, SCL
- MPU6050 : VCC, GND, SDA, SCL
- Brochage optimisé pour l’Arduino

Schéma de câblage



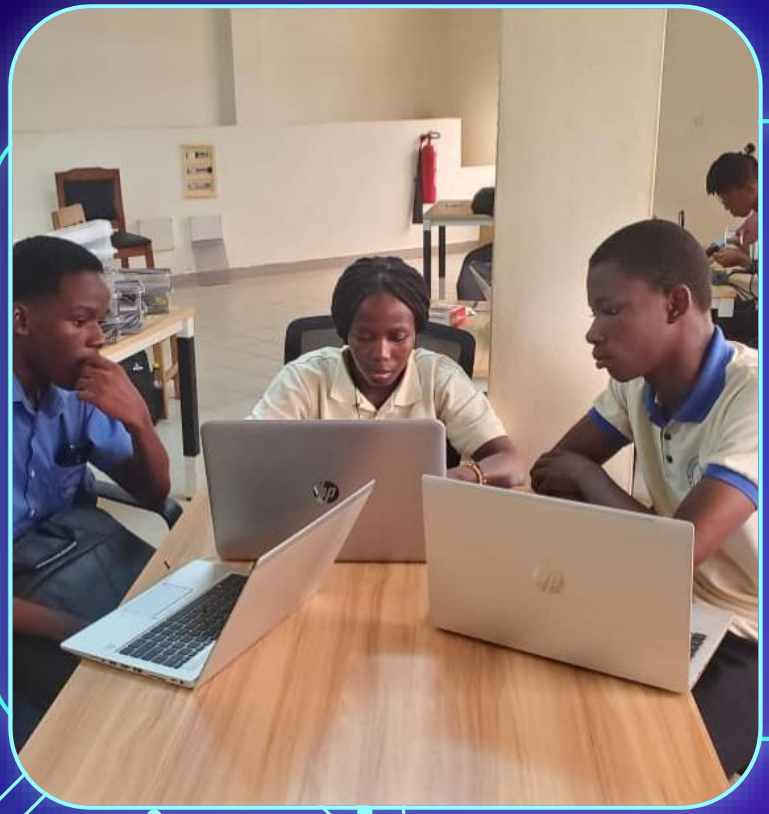
MONTAGE & IMPLÉMENTATION

Code Arduino

- Lecture des données du capteur
- Traitement (valeurs physiques, retrait gravité)
- Affichage : direction & accélération

Résultats

Le code lit les valeurs brutes d’accélération et de rotation, les convertit en unités physiques, puis retire la gravité pour isoler l’accélération réelle.
Le sens du mouvement est alors déduit et affiché en temps réel sur le LCD.



03. INFORMATIQUE

Réalisation d’un système de classe de gestion d’un Robot

PLAN

01 Présentation & Fonctionnalités

02 Diagramme de Classe

03 Implémentation en C++

Présentation & Fonctionnalité

Présentation

Notre robot autonome détecte, collecte et trie les déchets au sol tout en évitant les obstacles, sans intervention humaine.

Présentation & Fonctionnalités

Fonctionnalités

- Robot autonome pour environnement urbain
- Déplacement via véhicule robotique
- Détection intelligente des déchets
- Préhension et tri automatique avec bras robotique

Présentation & Fonctionnalité

Architecture matérielle

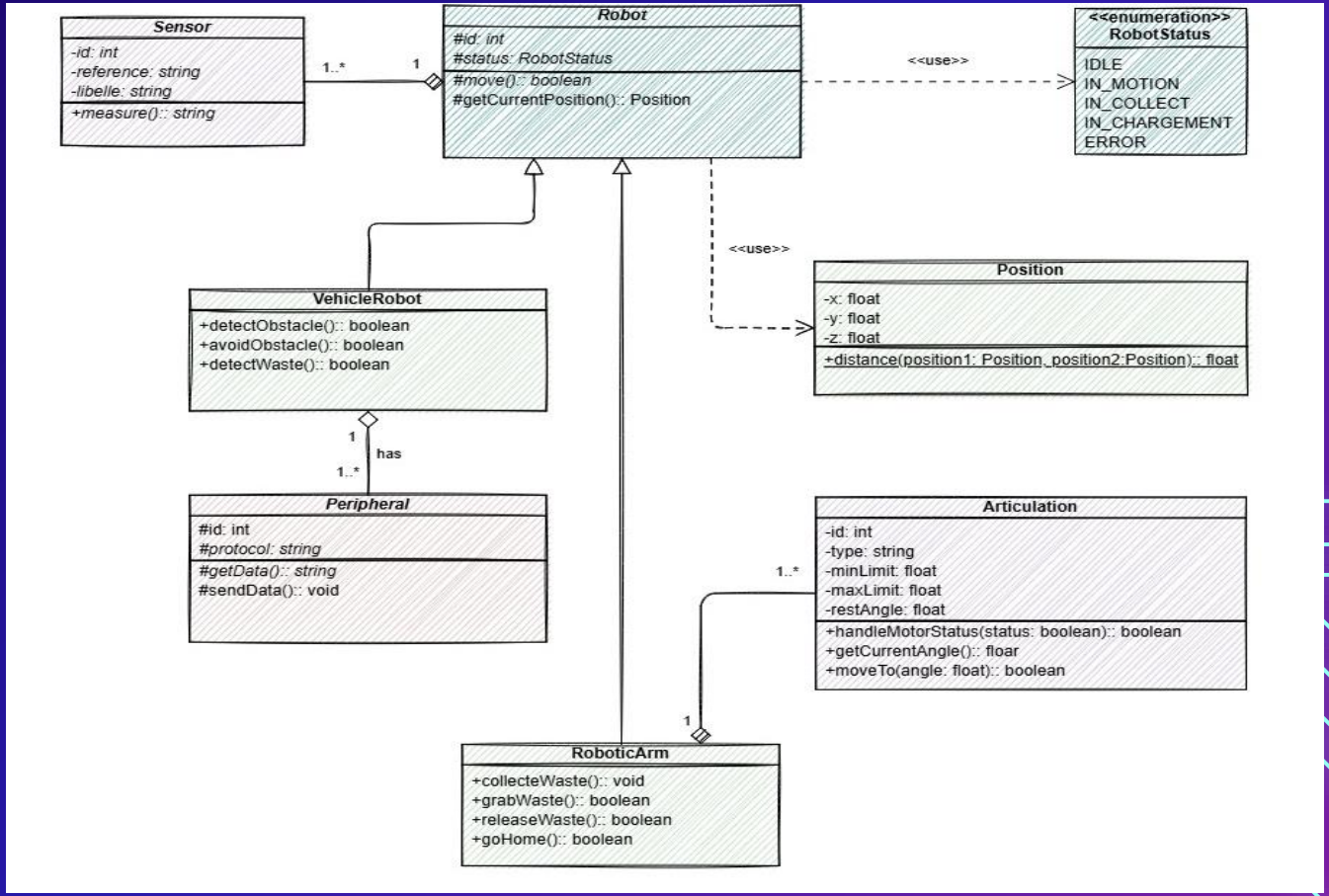
- **Rosmaster X3** pour le véhicule robotique
  - Navigation autonome (SLAM)
  - Détection d’obstacles (LiDAR)
  - Mobilité omnidirectionnelle (roues Mécanum)

Présentation & Fonctionnalité

Architecture matérielle

- **Jetson Nano (Dofbot)** pour le bras robotisé
  - Détection de déchet via camera
  - Classification par intelligence artificielle
  - Manipulation avec un bras 6 axes

Diagramme de Classe



Implémentation en C++

Classe robot

Robot.hpp

```cpp
class Robot {
protected:
    int id;
    RobotStatus status;
    std::vector<Sensor*> sensors;

    Robot(int id, RobotStatus status);

    virtual ~Robot();
    void addSensor(Sensor* sensor);
    virtual bool move() = 0;
    virtual Position getCurrentPosition() const = 0;

    // Getters et setters
    int getId() const { return id; }
    void setId(int val) { id = val; }
    RobotStatus getStatus() const { return status; }
    void setStatus(RobotStatus s) { status = s; }
    std::vector<Sensor*> getSensors() const { return sensors; }
};
```

Implémentation en C++

Classe RoboticArm

RoboticArm.hpp

```cpp
class RoboticArm : public Robot {
private:
    std::vector<Articulation*> articulations;
    Position position;
public:
    RoboticArm(int id, RobotStatus status);

    void addArticulation(Articulation* articulation);
    void collectWaste();
    bool grabWaste();
    bool releaseWaste();
    bool goHome();
    bool move() override;

    Position getCurrentPosition() const override;
    std::vector<Articulation*> getArticulations() const { return articulations; }
    void setArticulations(const std::vector<Articulation*>& a) { articulations = a;
}

    /**
     * @brief Accesseur pour la position.
     * @return Position actuelle.
     */
    Position getPosition() const { return position; }

    /**
     * @brief Mutateur pour la position.
     * @param pos Nouvelle position.
     */
    void setPosition(const Position& pos) { position = pos; }
};
```

Implémentation en C++

Classe VehiculeRobot

VehicleRobot.hpp

```
class VehicleRobot : public Robot {
private:
    std::vector<Peripheral*> peripherals;
    Position position;
public:
    VehicleRobot(int id, RobotStatus status);
    void addPeripheral(Peripheral* peripheral);
    bool detectObstacle();
    bool avoidObstacle();
    bool detectWaste();
    bool move() override;

    Position getCurrentPosition() const override;
    std::vector<Peripheral*> getPeripherals() const { return peripherals; }
    void setPeripherals(const std::vector<Peripheral*>& p) { peripherals = p; }
    Position getPosition() const { return position; }
    void setPosition(const Position& pos) { position = pos; }
};
```

Implémentation en C++

Move() classe VehiculeRobot

VehicleRobot.cpp

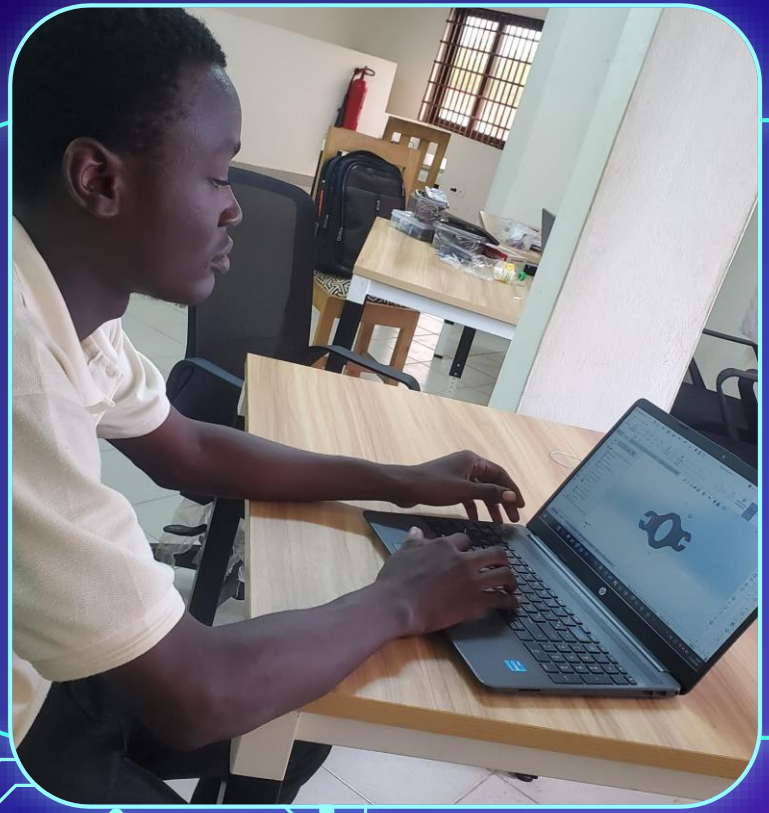
```cpp
bool VehicleRobot::move() {
    // Simuler le déplacement : avancer de 1 unité
    position.x += 1.0f;
    status = RobotStatus::IN_MOTION;
    // ...simuler le mouvement...
    if (detectObstacle()) {
        avoidObstacle();
    }
    status = RobotStatus::IDLE;
    return true;
}
```

Implémentation en C++

Move() classe RoboticArm

RoboticArm.cpp

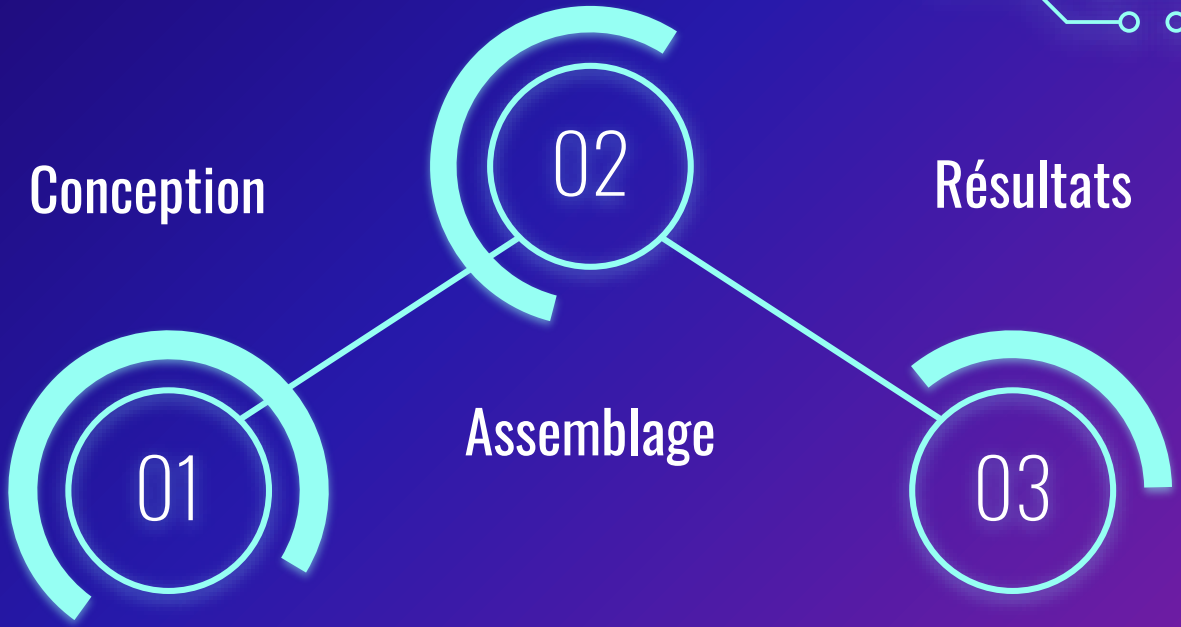
```cpp
bool RoboticArm::move() {
    // Simuler le déplacement vers une nouvelle position en utilisant les articulations
    bool success = true;
    for (auto* art : articulations) {
        float targetAngle = art->getCurrentAngle() + 10.0f;
        if (!art->moveTo(targetAngle)) success = false;
    }
    position.x += 1.0f;
    status = RobotStatus::IN_MOTION;
    // ...simuler le mouvement...
    // Déplacer chaque articulation vers un angle cible (simuler l'atteinte de la position cible)
    std::vector<float> targetAngles = {30.0f, 45.0f, 60.0f}; // Exemples d'angles cibles
    for (size_t i = 0; i < articulations.size() && i < targetAngles.size(); ++i) {
        if (!articulations[i]->moveTo(targetAngles[i])) success = false;
    }
    // Simuler la prise de déchets à la position cible
    if (!grabWaste()) success = false;
    // Déplacer vers la position de la poubelle (simuler avec de nouveaux angles)
    std::vector<float> trashAngles = {10.0f, 20.0f, 30.0f}; // Exemples d'angles pour la poubelle
    for (size_t i = 0; i < articulations.size() && i < trashAngles.size(); ++i) {
        if (!articulations[i]->moveTo(trashAngles[i])) success = false;
    }
    // Relâcher les déchets dans la poubelle
    if (!releaseWaste()) success = false;
    status = RobotStatus::IDLE;
    return success;
}
```



04. MECANIQUE

Niveau Débutant

PLAN



Présentation & Conception

Présentation

- Objectif : Concevoir un robot roulant capable de trier les déchets
- Rôle de la mécanique : conception des pièces, simulation des mouvements

Présentation & Conception

Conception

- **Pièce 1**
  - Étapes : esquisse de cercles, spline, extrusion, perçages
  - Matériau : AISI 1020
  - Images : IM 6 à IM 8
  - Détails : symétrie, coloration, masse

Présentation & Conception

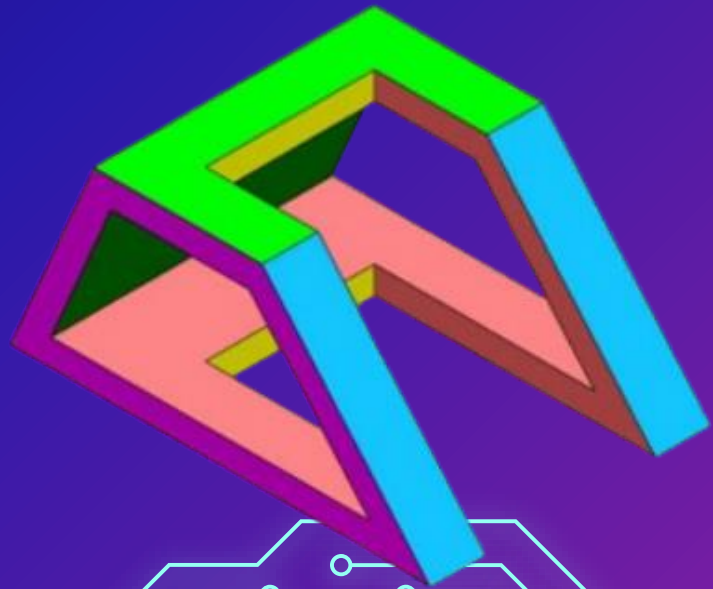
Conception

- **Pièce 2**
  - Étapes clés : cercles extrudés, perçages,
  - courbures, symétries
  - Matériau : Alliage d’aluminium 1060
  - Masse déterminée

Présentation & Conception

Conception

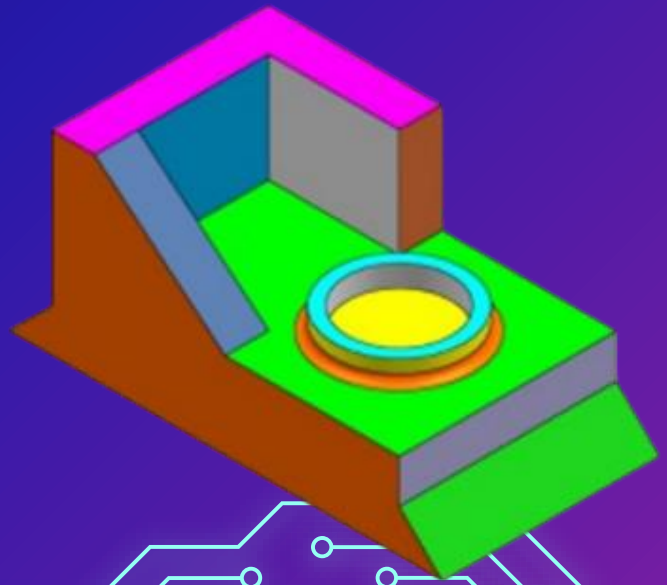
- **Pièce 3**
  - Processus : esquisse, décalage, extrusion, perçage
  - Matériau : acier AISI 1020
  - Masse calculée



Présentation & Conception

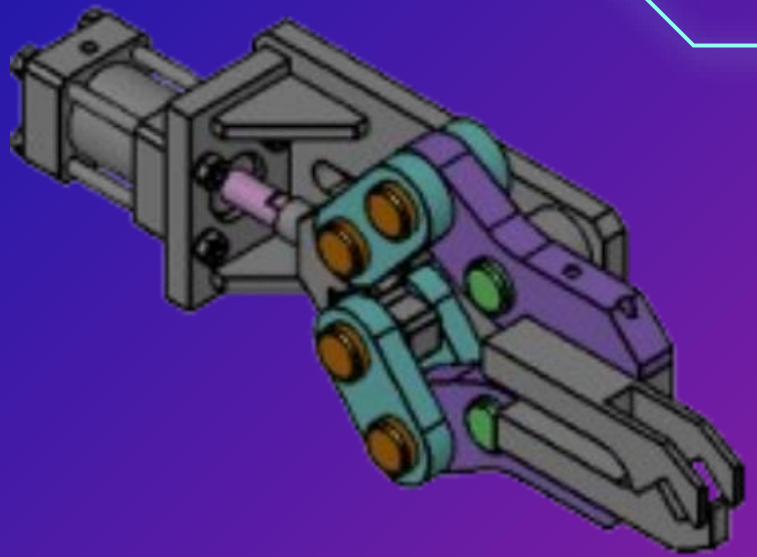
Conception

- **Pièce 4**
- Détails : extrusion, triangles, parallélogramme
- Images : image7 à image9, image 10
- Détails : extrusion d’anneaux, filetage, matériau : aluminium 1060



Assemblage

- Deux types de contraintes : vis/composants et composants entre eux



Résultats

Tableau des masses des pièces

Pièce	Masse (g)
Pièce 1	3020.54
Pièce 2	290.80
Pièce 3	1633.25
Pièce 4	297.29

Résultats

Centre de gravité de l’assemblage

Position du vérin	X (mm)	Y (mm)	Z (mm)
Position minimale	-25.78	0.06	19.81
Position maximale	-45.29	0.00	24.50

Conclusion

- ✓ Lecture et traitement de données capteurs (MPU6O.
- ✓ Programmation orientée objet en C++
- ✓ Architecture robotique modulaire (bras articulé + véhicule mobile).
- ✓ Conception mécanique intégrée et optimisée.

Merci !!

TEAM – IMSP