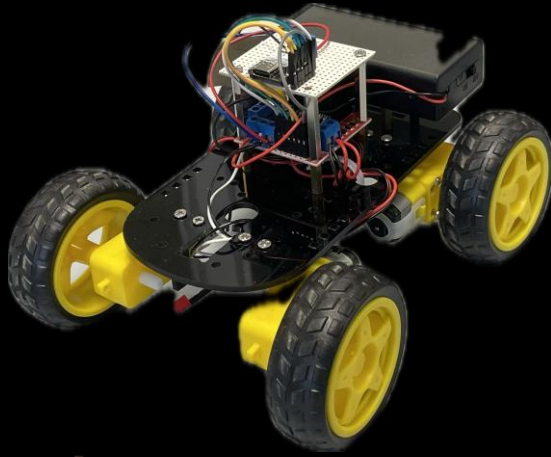
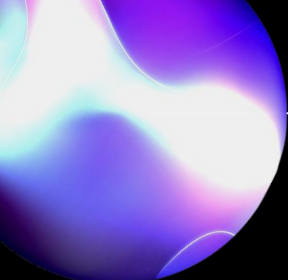






Sommaire :

01 Cahier des charges

02 Séquencement des
livrables

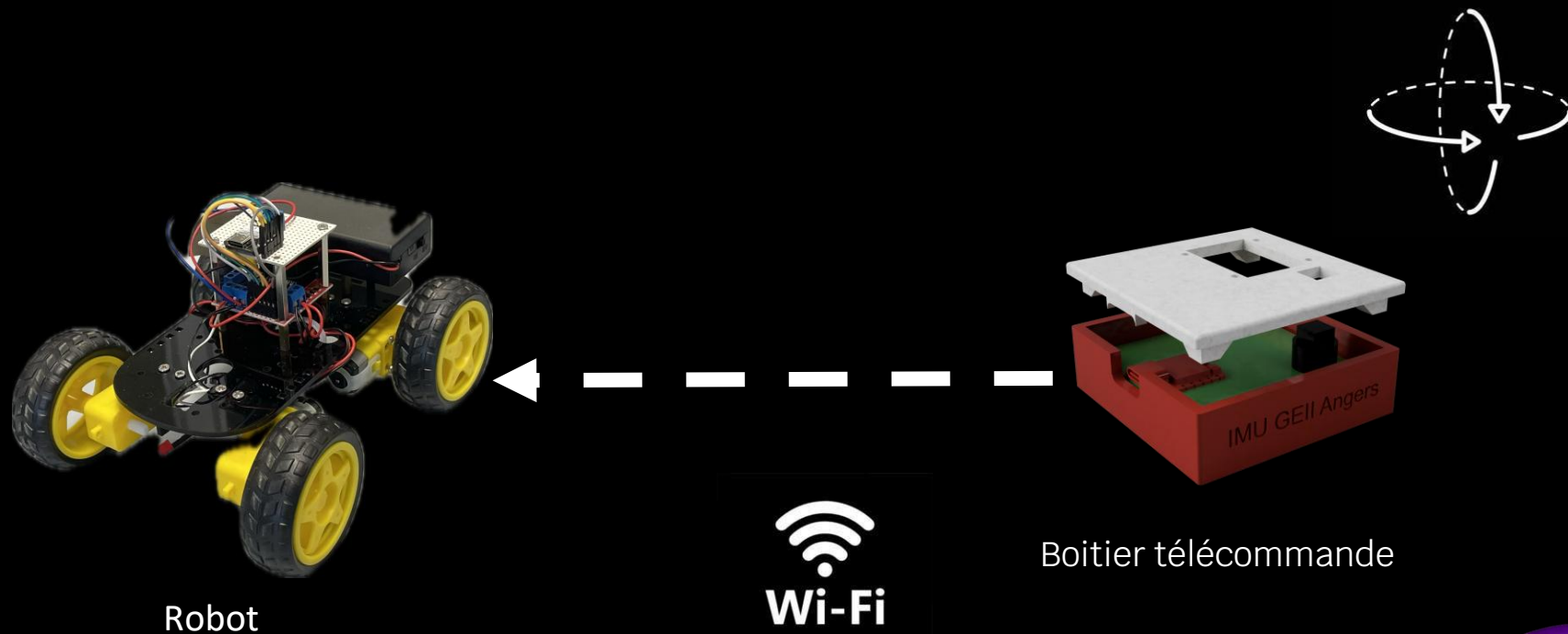
03 Choix des composants

04 Conception &
Programmation

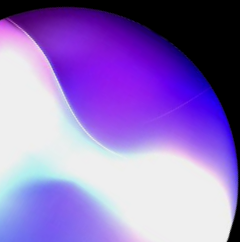
05 Réalisation

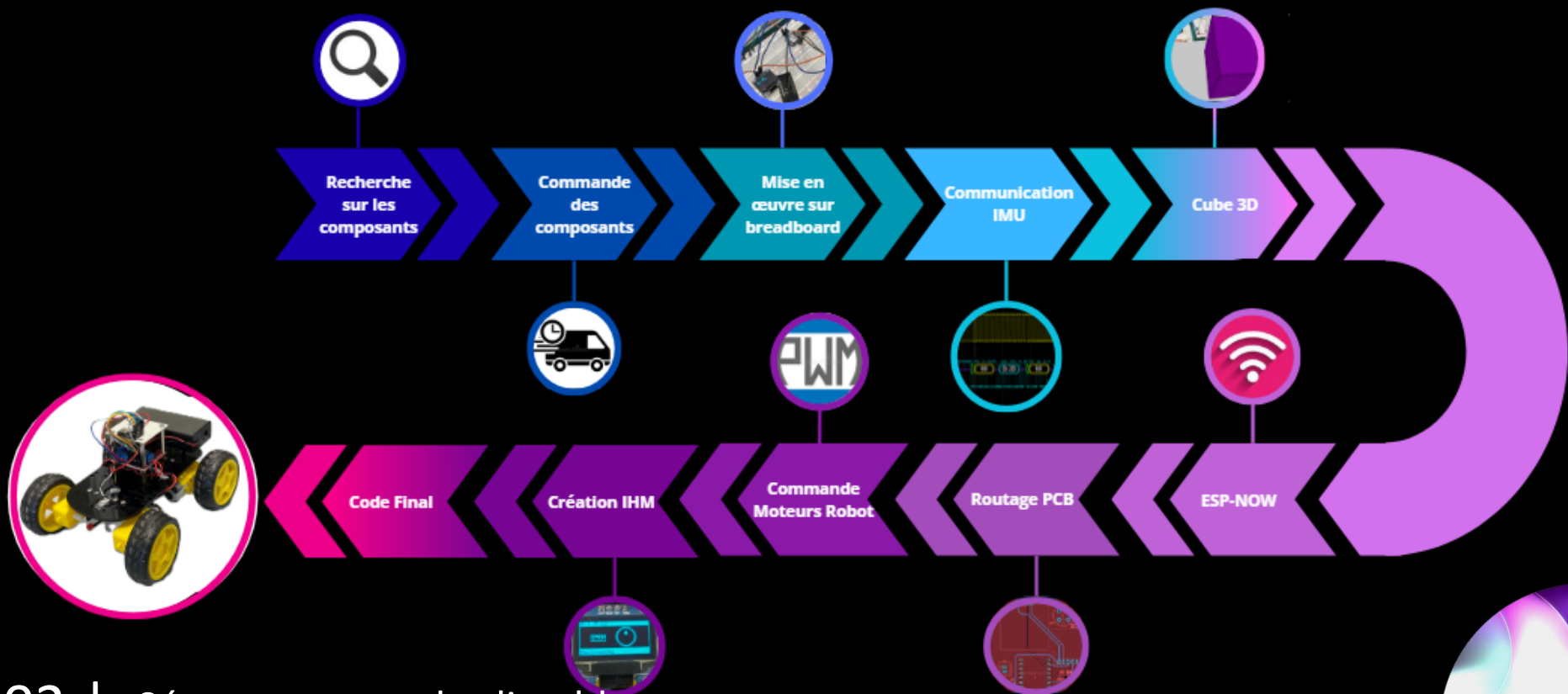
06 Conclusion & perspectives

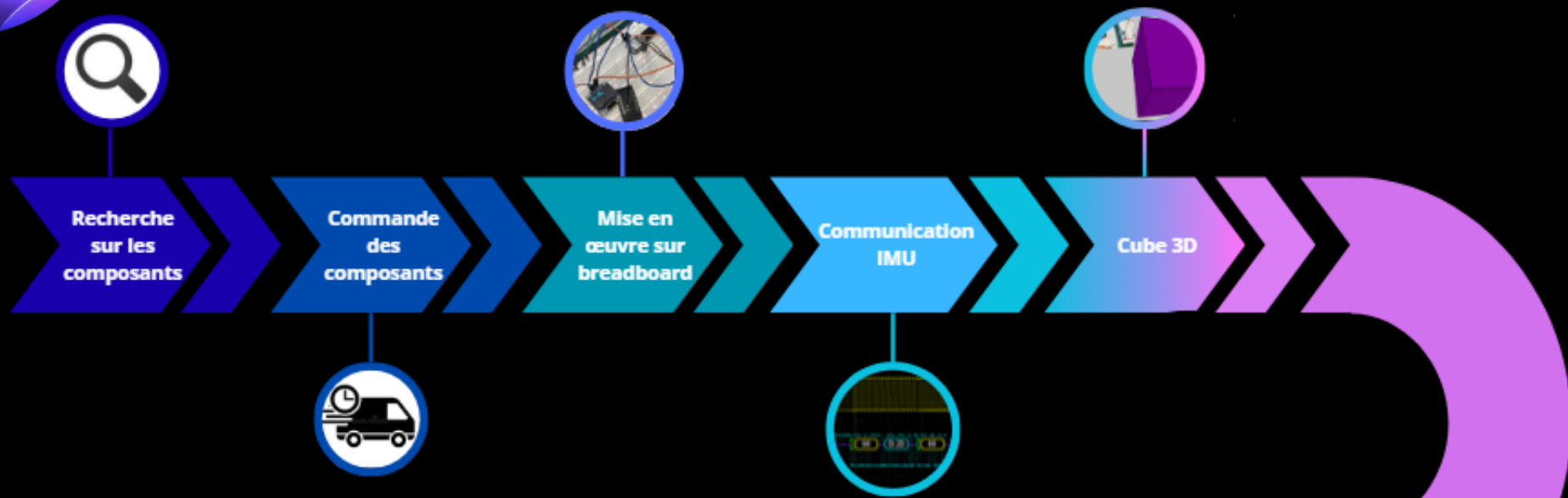
01 | Cahier des charges

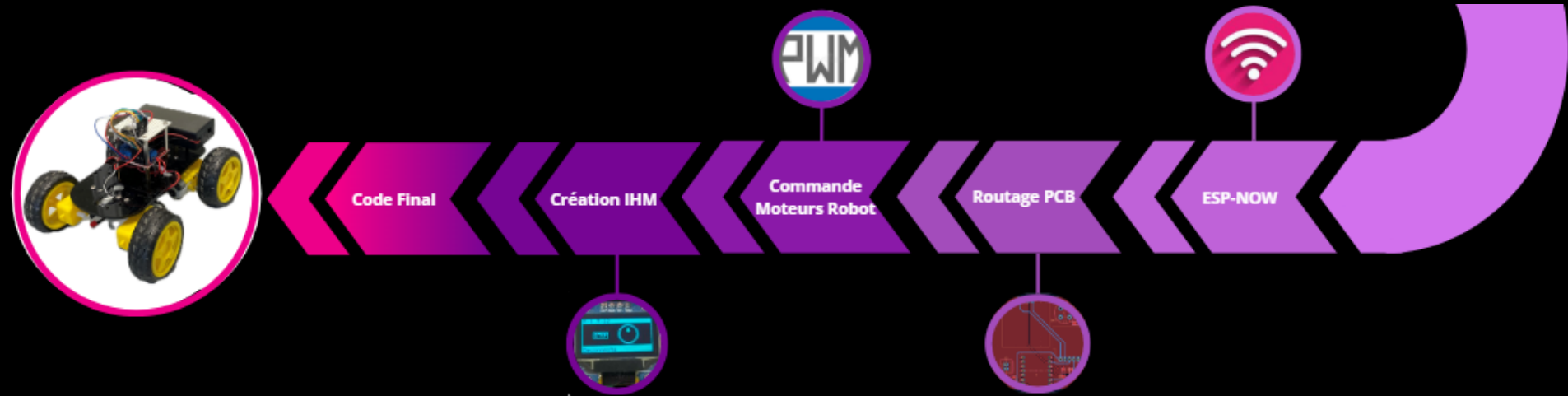


02 | Séquencement des livrables



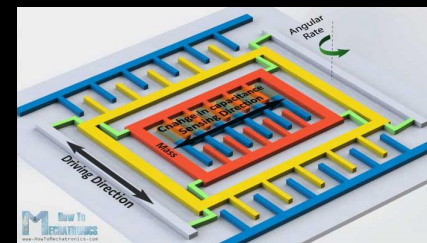
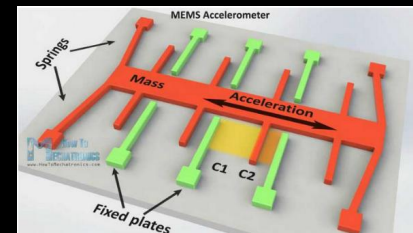
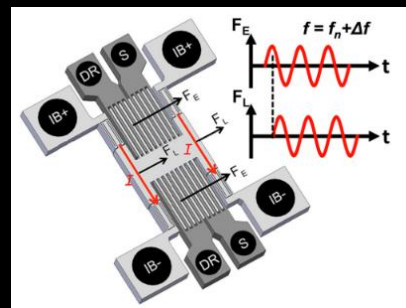


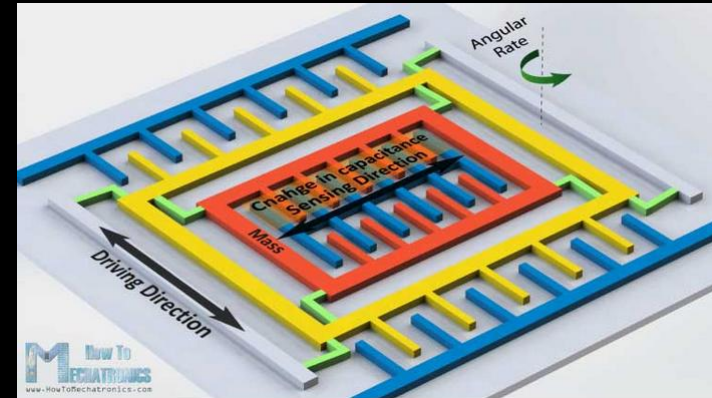
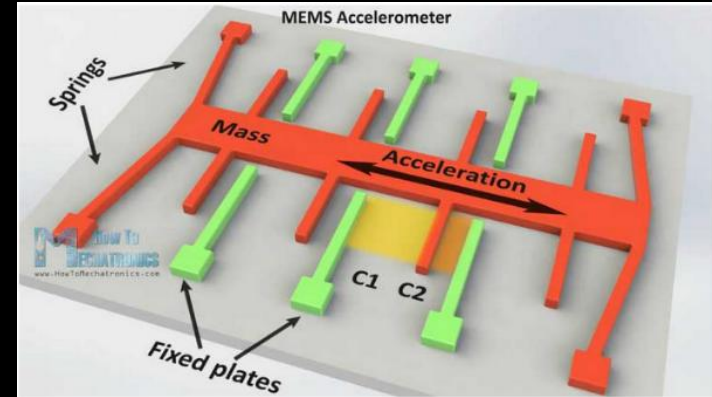
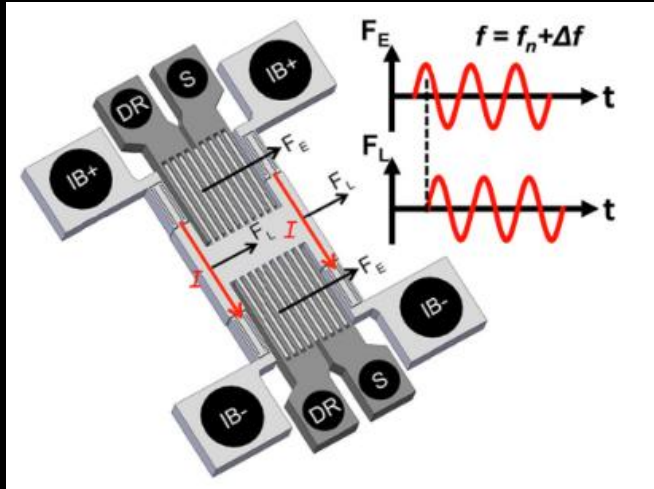




03 | Choix des composants

Choix IMU





Choix IMU :

ICM 20948



MPU-9250

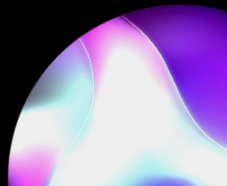


BNO055

Critère	ICM-20948	MPU-9250	BNO055
Prix	~17-25 €. (GO TRONIC 20,9 €.)	~8-20 €.	~30-45 €. (GO TRONIC 38,5 €.)
Qualité de mesure / bruit	Accéléromètre : densité de bruit $\approx 230 \mu\text{g}/\sqrt{\text{Hz}}$ Gyroscope : densité de bruit $\approx 0.017 \text{ }^\circ/\text{s}/\sqrt{\text{Hz}}$ Magnétomètre : résolution 16 bits, plage $\pm 4900 \mu\text{T}$	Accéléromètre : densité de bruit $\approx 300 \mu\text{g}/\sqrt{\text{Hz}}$, Gyroscope : densité de bruit $\approx 0.01 \text{ }^\circ/\text{s}/\sqrt{\text{Hz}}$	Accéléromètre : densité de bruit $\approx 170\text{--}190 \mu\text{g}/\sqrt{\text{Hz}}$ Gyroscope : bruit $\approx 0.014 \text{ }^\circ/\text{s}/\sqrt{\text{Hz}}$ Magnétomètre : offset calibré $\pm 2 \mu\text{T}$, précision capteur magnétique $\pm 2.5^\circ$
Consommation de courant	3.11mA	3.7mA	12.3mA

Choix IMU :

Critère	ICM-20948	MPU-9250	BNO055
Prix	~17-25 €. (GO TRONIC 20,9 €.)	~8-20 €.	~30-45 €. (GO TRONIC 38,5 €.)
Qualité de mesure / bruit	Accéléromètre : densité de bruit $\approx 230 \mu\text{g}/\sqrt{\text{Hz}}$ Gyroscope : densité de bruit $\approx 0.015 \text{ }^\circ/\text{s}/\sqrt{\text{Hz}}$ Magnétomètre : résolution 16 bits, plage $\pm 4900 \mu\text{T}$	Accéléromètre : densité de bruit $\approx 300 \mu\text{g}/\sqrt{\text{Hz}}$, Gyroscope : densité de bruit $\approx 0.01 \text{ }^\circ/\text{s}/\sqrt{\text{Hz}}$	Accéléromètre : densité de bruit $\approx 150\text{--}190 \mu\text{g}/\sqrt{\text{Hz}}$ Gyroscope : bruit $\approx 0.014 \text{ }^\circ/\text{s}/\sqrt{\text{Hz}}$
Consommation de courant	3.11mA	3.7mA	12.3mA



Choix Microcontrôleur :

Comparaison des microcontrôleurs :



ESP-32

Avantages :

- Wifi
- Bluetooth
- Faible consommation
- Bonne performance de calcul

Inconvénients :

- Complexité de configuration



Arduino

Avantages :

- Simplicité
- Pas chère
- Grand communauté

Inconvénients :

- Pas de Wifi
- Pas de Bluetooth
- Peu puissant
- Peu de mémoire



STM-32

Avantages :

- Très puissant
- Faible consommation d'énergie
- Large écosystème

Inconvénients :

- Pas de Wifi
- Pas de Bluetooth
- Complexité de configuration
- Cout élevé

Choix du type de Microcontrôleur :

Série	Architecture / Cœur	Wi-Fi / Bluetooth	Particularités	Consommation (typique)
ESP32-WROOM-32	Xtensa dual/single-core	Wi-Fi 2.4 GHz + Bluetooth classique & LE	Bonne puissance générale, large support	~160 mA (mode actif) / 10 μ A (veille)
ESP32-S2	Xtensa single-core	Wi-Fi 2.4 GHz (pas de Bluetooth)	Intègre USB OTG, consommation réduite	~160 mA (mode actif) / 20 μ A (veille)
ESP32-S3	Xtensa dual-core	Wi-Fi 2.4 GHz + Bluetooth LE	Support IA/ML, plus de GPIO disponibles	~240 mA (mode actif) / 10 μ A (veille)
ESP32-C3	RISC-V single-core	Wi-Fi 2.4 GHz + Bluetooth LE	Faible conso, sécurisé	~160 mA (mode actif) / 5 μ A (veille)
ESP32-C6	RISC-V single-core	Wi-Fi 6 + BLE + 802.15.4	Réseaux modernes (Thread, Zigbee), basse conso	~200 mA (mode actif) / 10 μ A (veille)
ESP32-H2	RISC-V single-core	BLE + 802.15.4 (pas de Wi-Fi)	Idéal réseaux maillés, sans-fil basse énergie	~100 mA (mode actif) / 1 μ A (veille)



ESP-32

Avantages :

- Wifi
- Bluetooth
- Faible consommation
- Bonne performance de calcul

Inconvénients :

- Complexité de configuration



Arduino

Avantages :

- Simplicité
- Pas chère
- Grand communauté

Inconvénients :

- Pas de Wifi
- Pas de Bluetooth
- Peu puissant
- Peu de mémoire



STM-32

Avantages :

- Très puissant
- Faible consommation d'énergie
- Large écosystème

Inconvénients :

- Pas de Wifi
- Pas de Bluetooth
- Complexité de configuration
- Cout élevé

Choix Microcontrôleur :

Comparaison des microcontrôleurs :



ESP-32

Avantages :

- Wifi
- Bluetooth
- Faible consommation
- Bonne performance de calcul

Inconvénients :

- Complexité de configuration



Arduino

Avantages :

- Simplicité
- Pas chère
- Grand communauté

Inconvénients :

- Pas de Wifi
- Pas de Bluetooth
- Peu puissant
- Peu de mémoire



STM-32

Avantages :

- Très puissant
- Faible consommation d'énergie
- Large écosystème

Inconvénients :

- Pas de Wifi
- Pas de Bluetooth
- Complexité de configuration
- Cout élevé

Choix du type Microcontrôleur :

Série	Architecture / Cœur	Wi-Fi / Bluetooth	Particularités	Consommation (typique)
ESP32-WROOM-32	Xtensa dual/single-core	Wi-Fi 2.4 GHz + Bluetooth classique & LE	Bonne puissance générale, large support	~160 mA (mode actif) / 10 µA (veille)
ESP32-S2	Xtensa single-core	Wi-Fi 2.4 GHz (pas de Bluetooth)	Intègre USB OTG, consommation réduite	~160 mA (mode actif) / 20 µA (veille)
ESP32-S3	Xtensa dual-core	Wi-Fi 2.4 GHz + Bluetooth LE	Support IA/ML, plus de GPIO disponibles	~240 mA (mode actif) / 10 µA (veille)
ESP32-C3	RISC-V single-core	Wi-Fi 2.4 GHz + Bluetooth LE	Faible conso, sécurisé	~160 mA (mode actif) / 5 µA (veille)
ESP32-C6	RISC-V single-core	Wi-Fi 6 + BLE + 802.15.4	Réseaux modernes (Thread, Zigbee), basse conso	~200 mA (mode actif) / 10 µA (veille)
ESP32-H2	RISC-V single-core	BLE + 802.15.4 (pas de Wi-Fi)	Idéal réseaux maillés, sans-fil basse énergie	~100 mA (mode actif) / 1 µA (veille)

Série	Architecture / Cœur	Wi-Fi / Bluetooth	Particularités	Consommation (typique)
ESP32- WROOM-32	Xtensa dual/single-core	Wi-Fi 2.4 GHz + Bluetooth classique & LE	Bonne puissance générale, large support	160 mA (mode actif) / 10 μ A (veille)
ESP32-S2	Xtensa single-core	Wi-Fi 2.4 GHz (pas de Bluetooth)	Intègre USB OTG, consommation réduite	160 mA (mode actif) / 20 μ A (veille)
ESP32-S3	Xtensa dual-core	Wi-Fi 2.4 GHz + Bluetooth LE	Support IA/ML, plus de GPIO disponibles	240 mA (mode actif) / 10 μ A (veille)
ESP32-C3	RISC-V single-core	Wi-Fi 2.4 GHz + Bluetooth LE	Faible conso, sécurisé	160 mA (mode actif) / 5 μ A (veille)
ESP32-C6	RISC-V single-core	Wi-Fi 6 + BLE + 802.15.4	Réseaux modernes (Thread, Zigbee), basse conso	200 mA (mode actif) / 10 μ A (veille)
ESP32-H2	RISC-V single-core	BLE + 802.15.4 (pas de Wi-Fi)	Idéal réseaux maillés, sans-fil basse énergie	100 mA (mode actif) / 1 μ A (veille)

04 | Conception & Programmation

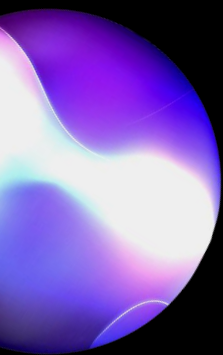
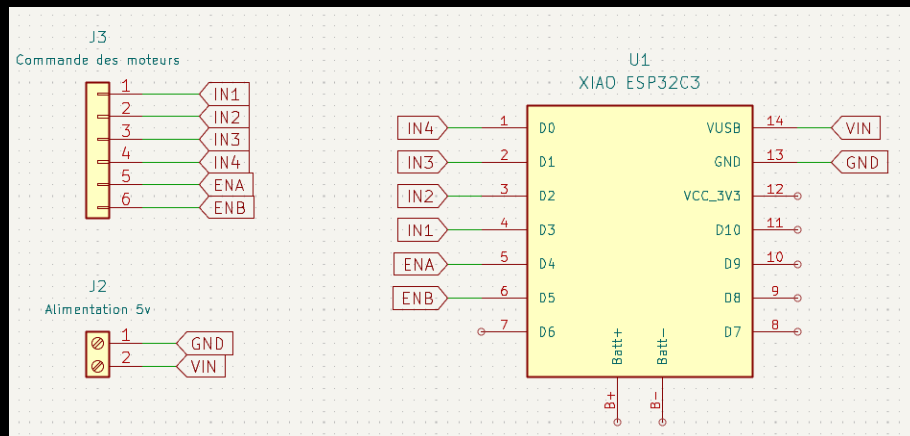
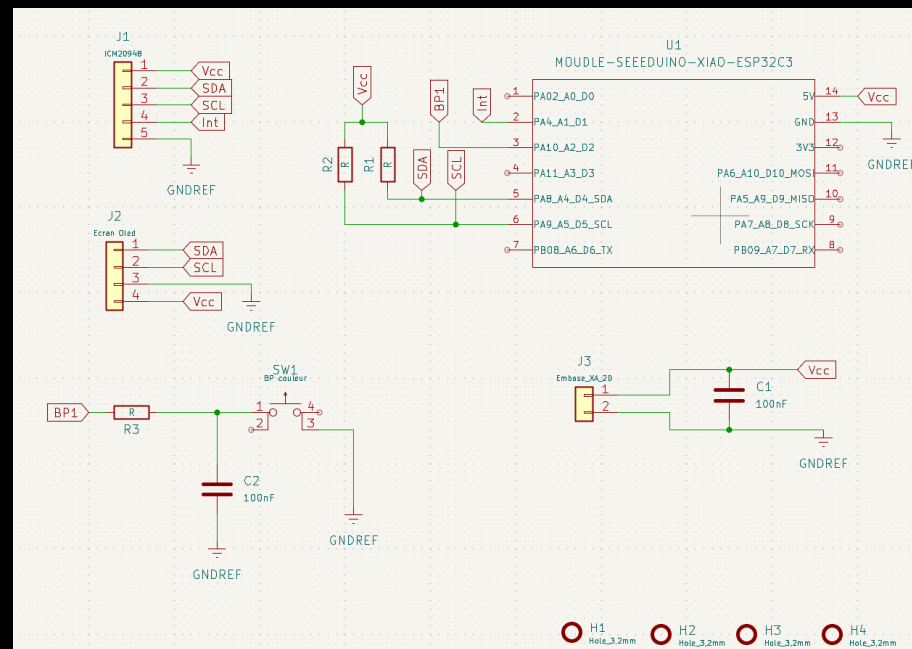


Schéma structurel :

Robot :

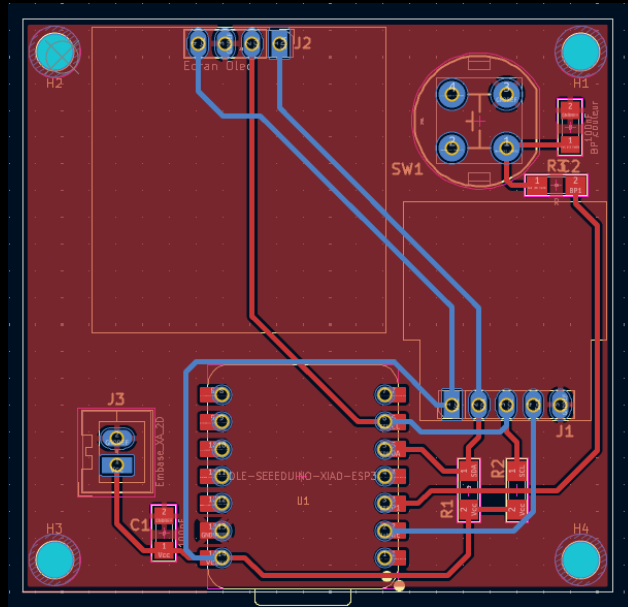


Télécommande :

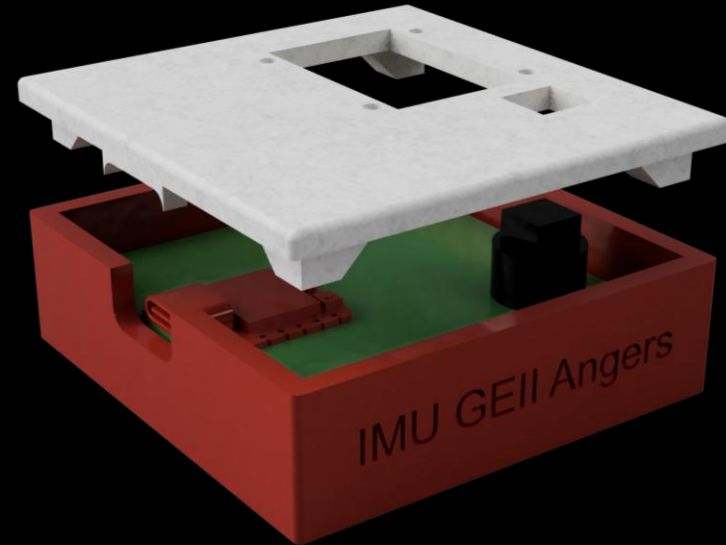


Carte électronique (PCB) :

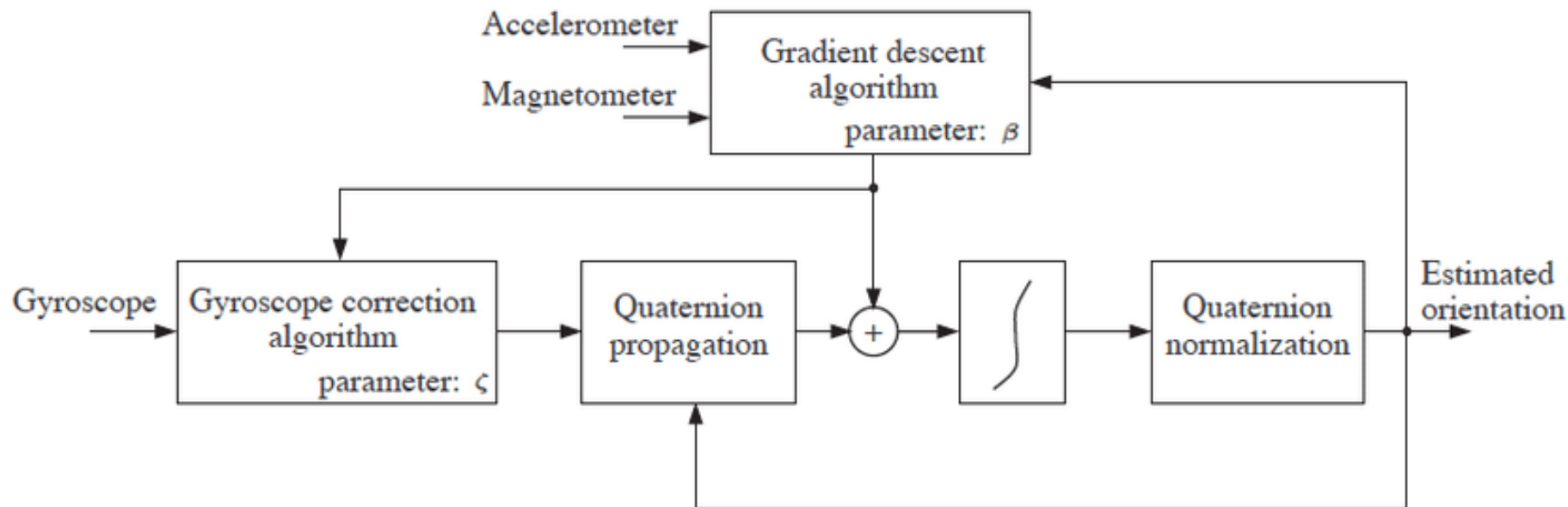
Routage du PCB partie Télécommande :



Boîtier imprimé en 3D destiné au PCB



Programmation :





Programmation :

```
#include <ICM20948_WE.h>
```

```
float tangage = imu.getPitch();
float roulis = imu.getRoll();
imu.readSensor();
```

```
// IMU
imu.init();
imu.enableAcc(true);
imu.enableGyr(true);
imu.setAccRange(ICM20948_ACC_RANGE_2G);
imu.setGyrRange(ICM20948_GYRO_RANGE_250);
selectionBanque(0);

// --- configuration DLPF et fréquence ---
// Gyro : DLPF OFF, SMPLRT_DIV = 0 -> ~9 kHz
myIMU.setGyrDLPF(ICM20948_DLPF_OFF); // pas de filtre numérique gyro
myIMU.setGyrSampleRateDivider(0); // diviseur 0 => fréquence max (~9 kHz)

// Accélération : DLPF OFF, SMPLRT_DIV = 0 -> ~4,5 kHz
myIMU.setAccDLPF(ICM20948_DLPF_OFF); // pas de filtre numérique accel
myIMU.setAccSampleRateDivider(0); // diviseur 0 => fréquence max (~4,5 kHz)
```

Programmation :

```
imu.readSensor();
```

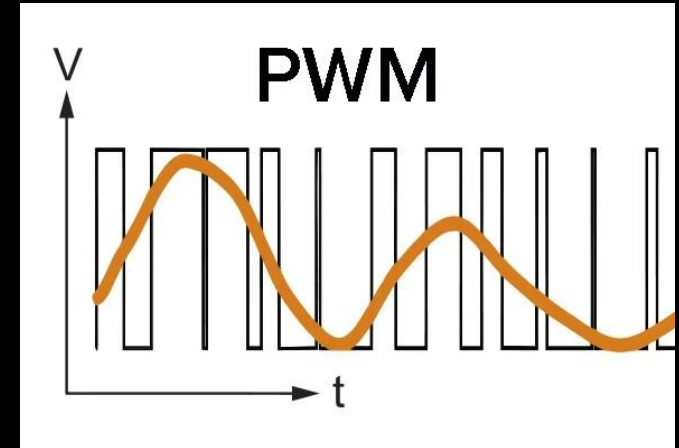
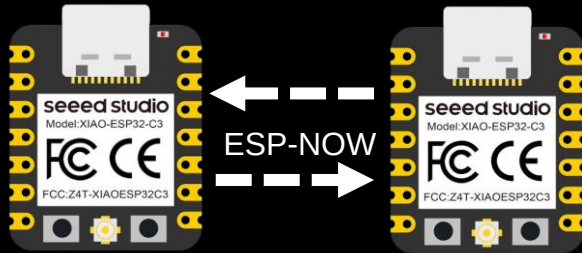
```
// Lecture accel + gyro bruts
void readAccelGyro(int16_t *accel, int16_t *gyro) {
    setBank(0);
    uint8_t buf[12];
    readBytes(ICM20948_ADDR, ACCEL_XOUT_H, 12, buf);

    accel[0] = ((int16_t)buf[0] << 8) | buf[1];
    accel[1] = ((int16_t)buf[2] << 8) | buf[3];
    accel[2] = ((int16_t)buf[4] << 8) | buf[5];
    gyro[0] = ((int16_t)buf[6] << 8) | buf[7];
    gyro[1] = ((int16_t)buf[8] << 8) | buf[9];
    gyro[2] = ((int16_t)buf[10] << 8) | buf[11];
}
```

Programmation :

```
#include <esp_now.h>
```

```
#include <L298N.h>
```



Programmation (IHM) :

HUD d'un avion :



IHM du robot :

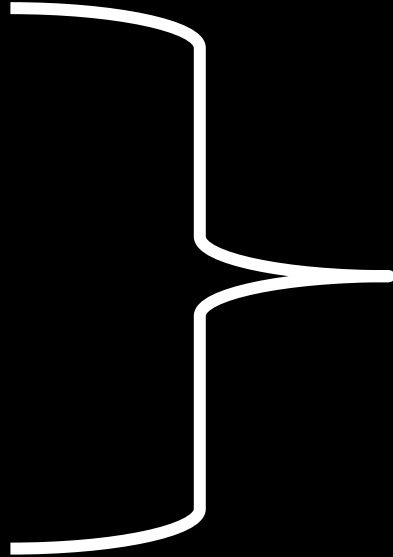
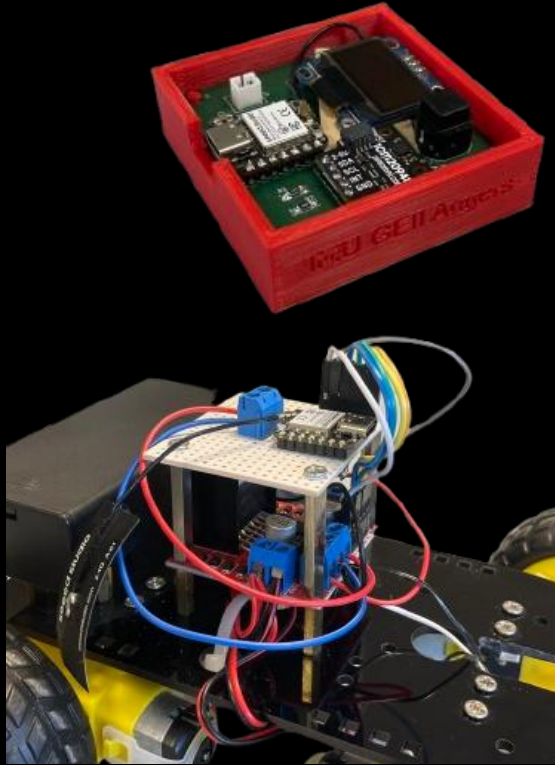


Claude

```
#include <Adafruit_SSD1306.h>
```

```
ecran.fillRect(0, 0, 128, 10, SSD1306_WHITE);
ecran.drawLine(x, y, x, y, SSD1306_BLACK);
ecran.setCursor(3, 2);
ecran.setTextColor(SSD1306_BLACK);
ecran.print("R:");
ecran.display();
```

05 | Réalisation



06 | Conclusion & perspectives

Perspectives :



Détection de mouvement
Commande dynamique
Enregistrement de commande

Merci de votre attention

Des questions ?