

## Test 1 électronique:

# Détection du Mouvement et de l'Orientation de la Main à l'aide d'un Capteur Gyroscope-Accéléromètre I2C

Réalisé par:

BENZAGHAR HIBA

EL HATIMY MARYAM

EL GOUMRI HIBA

Encadré par :

Yannis BORNA ( mentor électronique)

# Plan

▶▶▶	Introduction	01
▶▶▶	Choix du capteur	02
▶▶▶	Principe de fonctionnement	03
▶▶▶	Schéma électronique – KICAD	04
▶▶▶	Implémentation matérielle	05
▶▶▶	Code Arduino	06
▶▶▶	Vidéo de démonstration	07





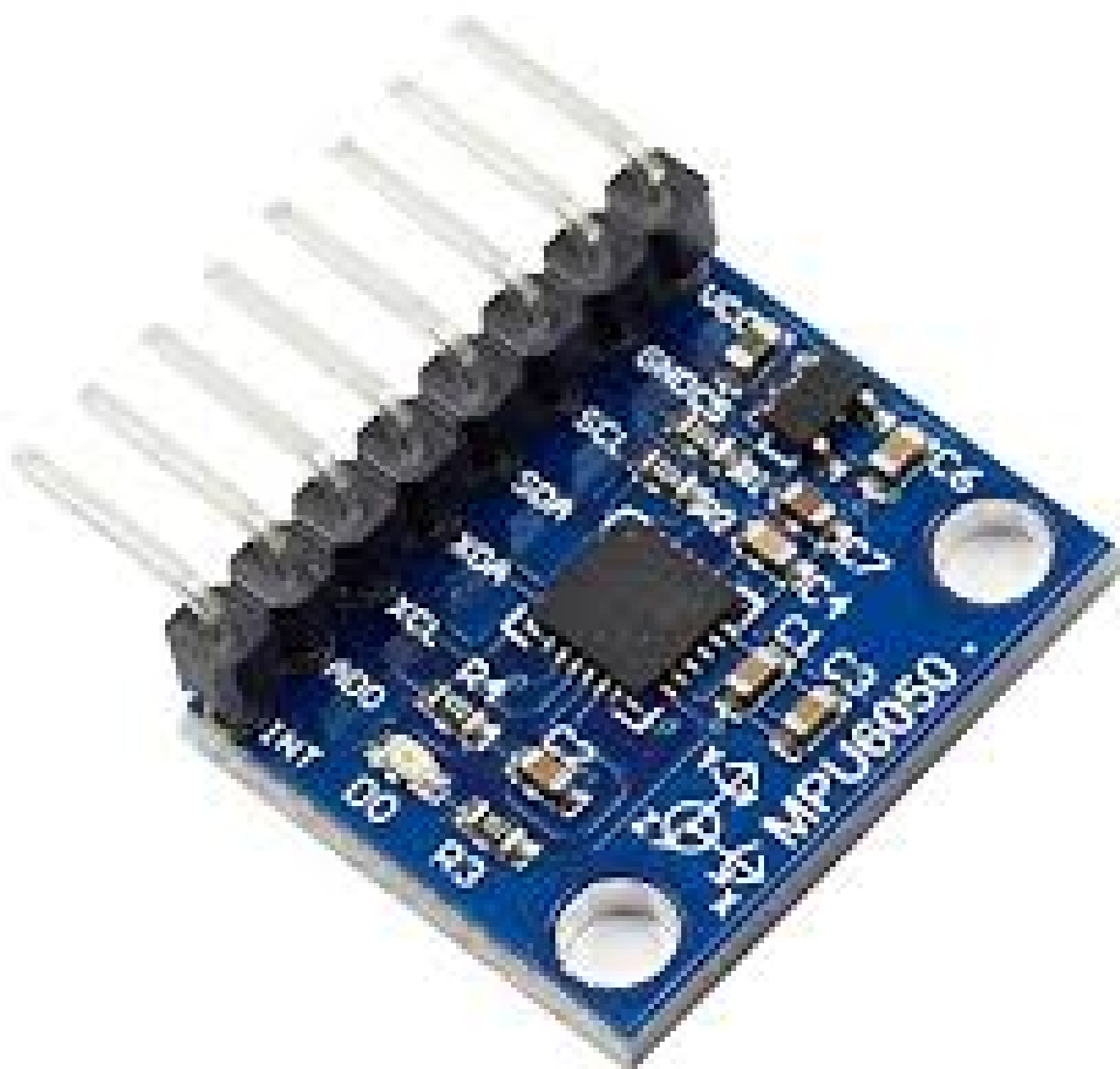
# Introduction

Dans le cadre du challenge TRC2K25, ce projet vise à développer un système embarqué capable d'exploiter les données d'un capteur combinant gyroscope et accéléromètre, communiquant via le protocole I2C. L'objectif est de détecter les mouvements de la main dans différentes directions et d'afficher en temps réel le sens du déplacement ainsi que l'accélération sur un écran LCD.

Cette réalisation implique la sélection d'un capteur adapté, la conception d'un montage électronique, la programmation d'un microcontrôleur Arduino, ainsi que la mise en œuvre d'une alimentation autonome.



# Pourquoi avons-nous choisi le capteur MPU6050 ?

**01****Fonctionnalité complète**

Le MPU6050 combine un accéléromètre 3 axes et un gyroscope 3 axes dans un seul module compact, ce qui permet de mesurer à la fois l'orientation et la vitesse de déplacement de la main avec précision

**02****Communication I2C facile à intégrer**

Il utilise une interface I2C, parfaitement compatible avec les microcontrôleurs comme l'Arduino, ce qui facilite la lecture des données avec un minimum de fils et une consommation réduite.

**03****Facilité d'intégration logicielle**

Il est largement utilisé dans la communauté Arduino, avec de nombreuses bibliothèques disponibles (comme MPU6050.h ou Wire.h), ce qui simplifie le développement et le débogage.



# Principe de fonctionnement

Le capteur MPU6050 combine deux fonctions principales : un accéléromètre et un gyroscope. L'accéléromètre permet de mesurer l'accélération linéaire sur les trois axes (X, Y, Z), ce qui permet de détecter les mouvements de la main vers le haut, le bas, la gauche, la droite, l'avant ou l'arrière. De son côté, le gyroscope mesure la vitesse angulaire sur ces mêmes axes, ce qui permet de connaître l'orientation de la main, comme une inclinaison ou une rotation.

Pour transmettre les données au microcontrôleur, le MPU6050 utilise le protocole de communication I2C. Ce protocole est simple et efficace, car il ne nécessite que deux fils : la ligne SCL (pour l'horloge) et la ligne SDA (pour les données). Grâce à cette liaison, le microcontrôleur peut envoyer des commandes au capteur et lire ses valeurs en temps réel.



01

Comment le MPU6050 envoie-t-il les données ?

Pour que le MPU6050 envoie les données, il ne suffit pas de simplement le connecter. Étant donné qu'on utilise une communication I2C, il faut suivre un protocole bien défini :

- **Envoi d'une commande au capteur**

Avant de lire une donnée, l'Arduino doit envoyer l'adresse du registre souhaité. Cette adresse indique quel type de donnée on veut (accélération, vitesse angulaire).

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]				
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]				
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]				
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]				
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]				
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]				

43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]				
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]				
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]				
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]				
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]				
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]				



## Structure des données reçues

02

Chaque mesure (accélération ou rotation) est codée sur 16 bits (2 octets) :

- **8 bits MSB (poids fort)**
- **8 bits LSB (poids faible)**

Les valeurs sont en complément à deux (signed integer), ce qui permet d'indiquer un sens positif ou négatif du mouvement.



## Structure des données reçues

02

Chaque mesure (accélération ou rotation) est codée sur 16 bits (2 octets) :

- **8 bits MSB (poids fort)**
- **8 bits LSB (poids faible)**

Les valeurs sont en complément à deux (signed integer), ce qui permet d'indiquer un sens positif ou négatif du mouvement.

## Conversion des données brutes

03

Pour exploiter les mesures, il faut les convertir en unités physiques :

- Accélération en g (gravité terrestre)
- Rotation en °/s (degrés par seconde)





## Structure des données reçues

02

Chaque mesure (accélération ou rotation) est codée sur 16 bits (2 octets) :

- **8 bits MSB (poids fort)**
- **8 bits LSB (poids faible)**

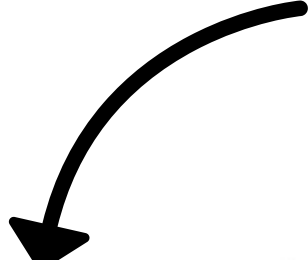
Les valeurs sont en complément à deux (signed integer), ce qui permet d'indiquer un sens positif ou négatif du mouvement.

## Conversion des données brutes

03

Pour exploiter les mesures, il faut les convertir en unités physiques :

- Accélération en g (gravité terrestre)
- Rotation en °/s (degrés par seconde)



AFS_SEL	Full Scale Range	LSB Sensitivity
0	$\pm 2g$	16384 LSB/g
1	$\pm 4g$	8192 LSB/g
2	$\pm 8g$	4096 LSB/g
3	$\pm 16g$	2048 LSB/g



## Structure des données reçues

**02**

Chaque mesure (accélération ou rotation) est codée sur 16 bits (2 octets) :

- **8 bits MSB (poids fort)**
- **8 bits LSB (poids faible)**

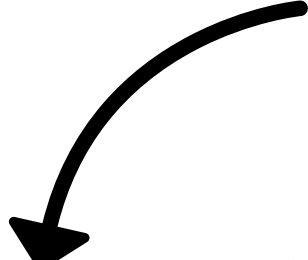
Les valeurs sont en complément à deux (signed integer), ce qui permet d'indiquer un sens positif ou négatif du mouvement.

## Conversion des données brutes

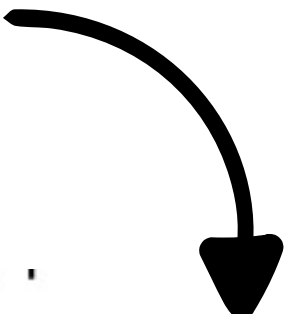
**03**

Pour exploiter les mesures, il faut les convertir en unités physiques :

- Accélération en g (gravité terrestre)
- Rotation en °/s (degrés par seconde)



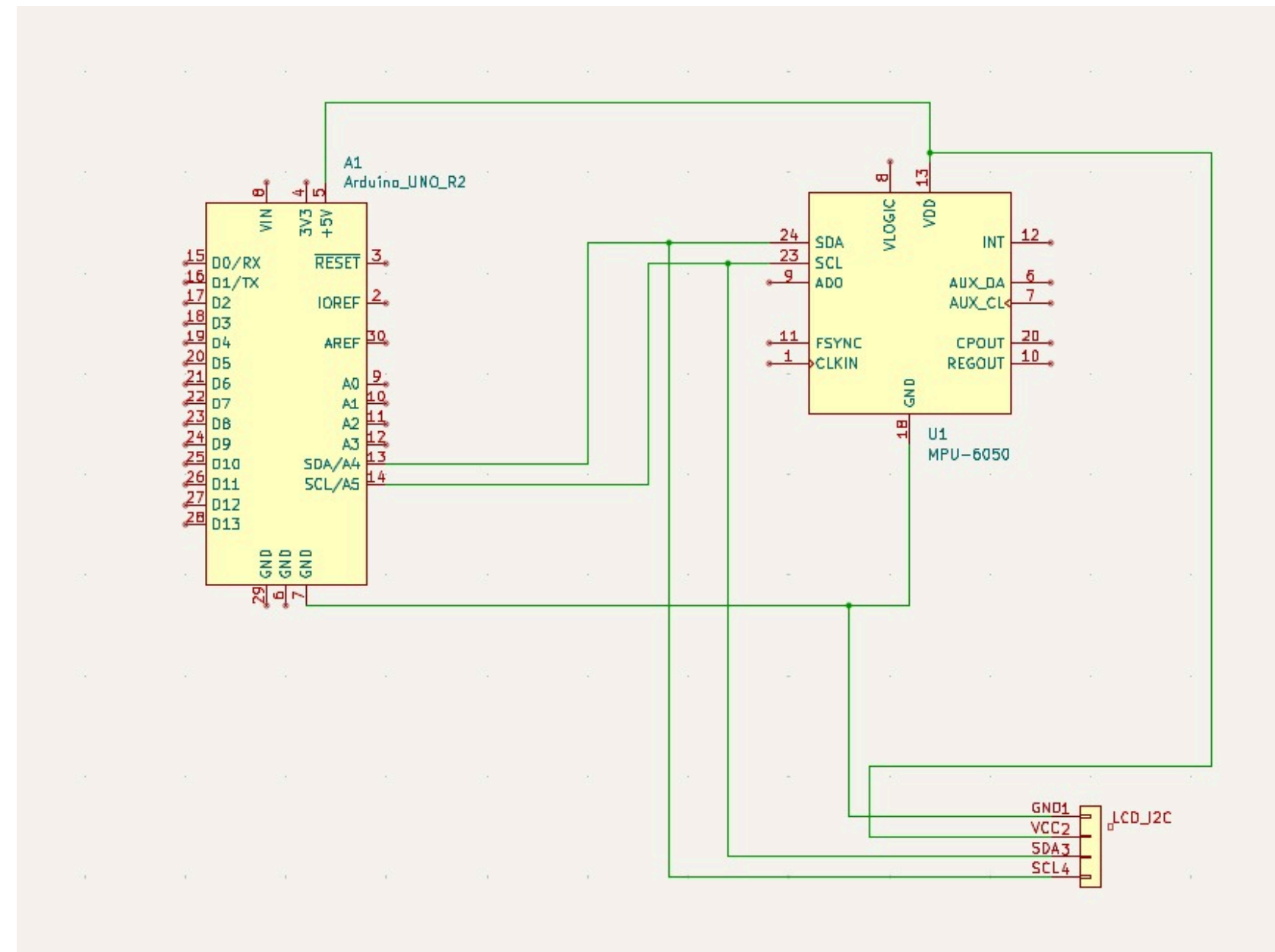
AFS_SEL	Full Scale Range	LSB Sensitivity
0	$\pm 2g$	16384 LSB/g
1	$\pm 4g$	8192 LSB/g
2	$\pm 8g$	4096 LSB/g
3	$\pm 16g$	2048 LSB/g



FS_SEL	Full Scale Range	LSB Sensitivity
0	$\pm 250$ °/s	131 LSB/°/s
1	$\pm 500$ °/s	65.5 LSB/°/s
2	$\pm 1000$ °/s	32.8 LSB/°/s
3	$\pm 2000$ °/s	16.4 LSB/°/s



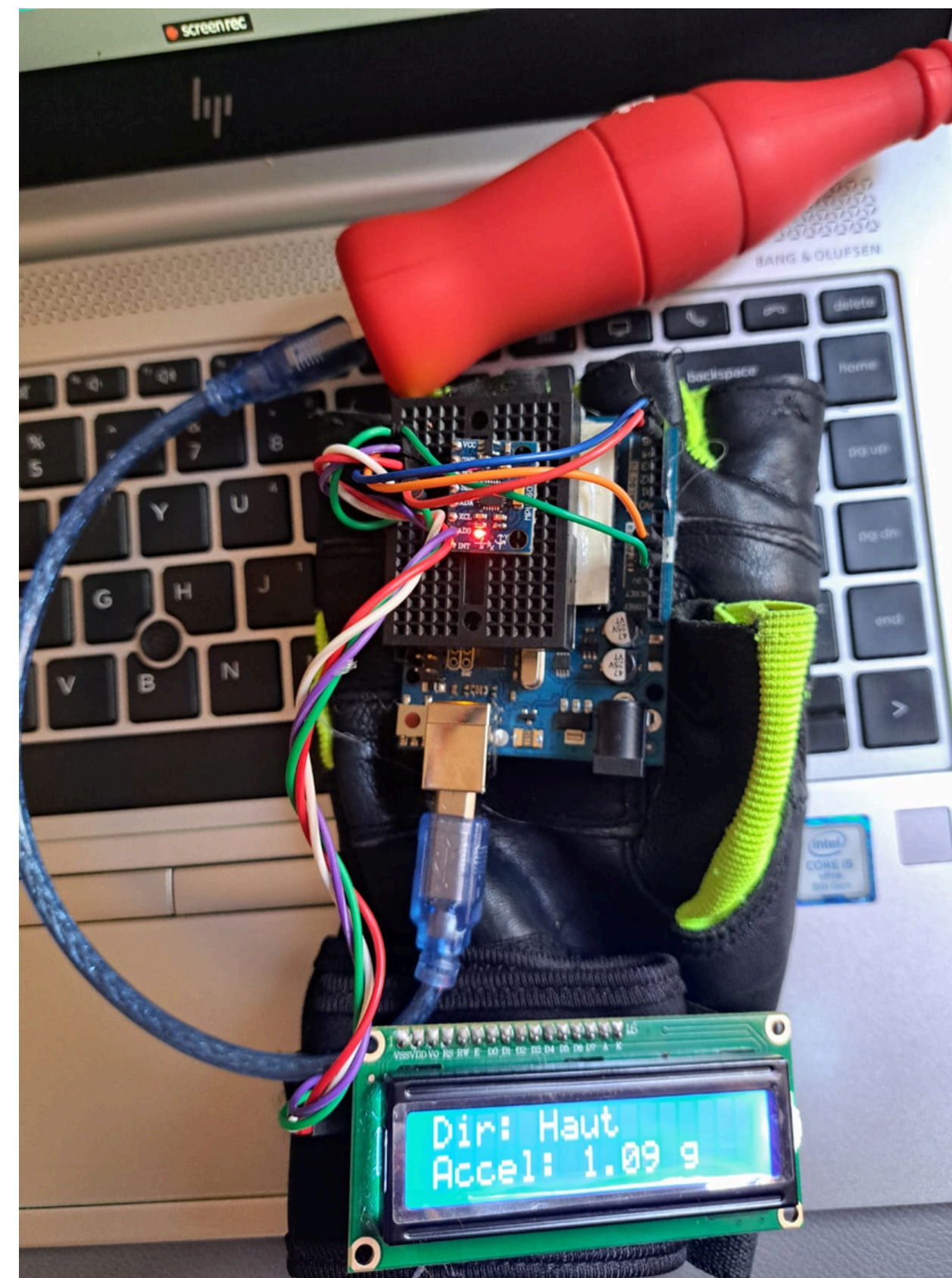
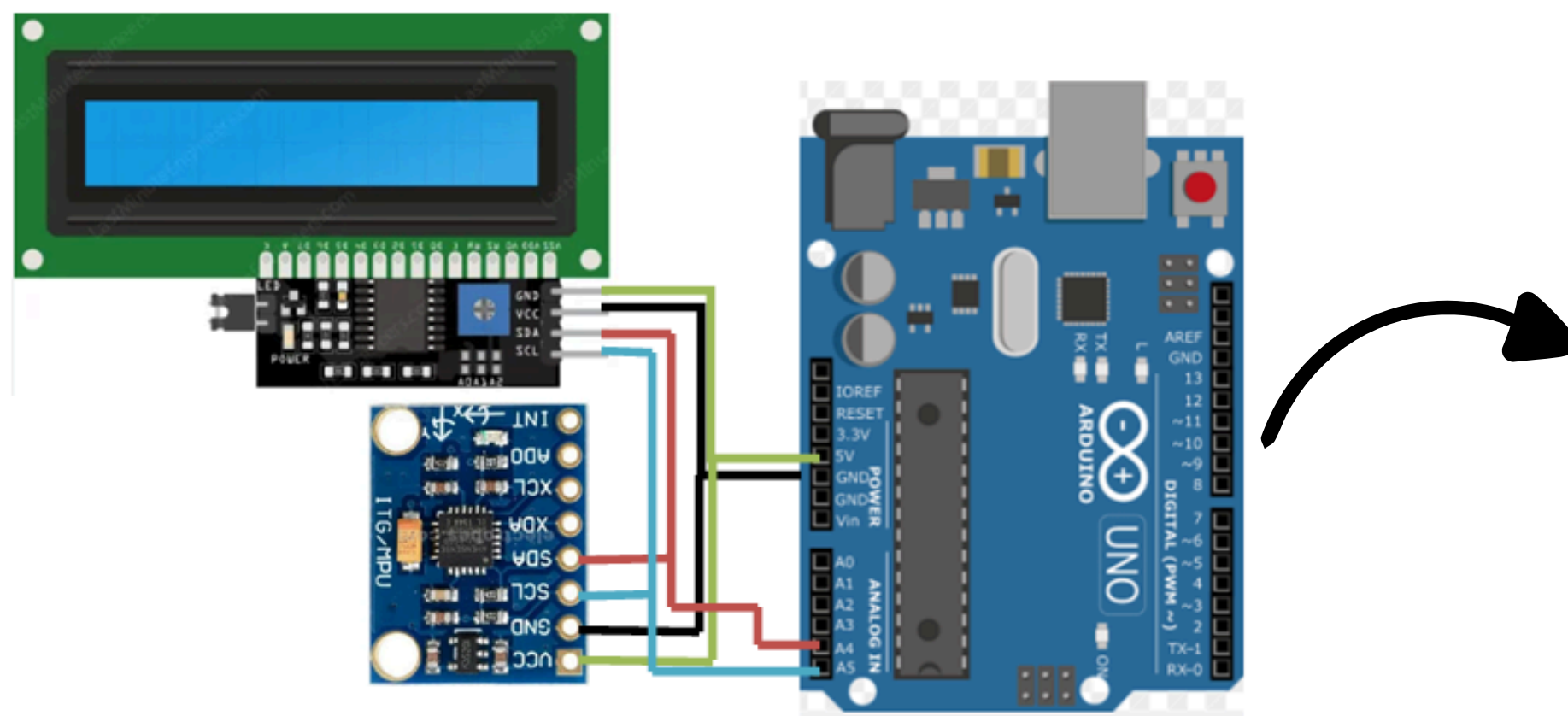
# Schéma électronique sous KICAD







# Implémentation matérielle





# Code Arduino

```
sketch_jun11a.ino
1  #include <Wire.h>           // Bibliothèque pour communication I2C
2  #include <LiquidCrystal_I2C.h> // Bibliothèque pour écran LCD I2C
3  #include <MPU6050.h>        // Bibliothèque pour capteur MPU6050
4  // Initialisation écran LCD avec adresse I2C 0x27, 16 colonnes et 2 lignes
5  LiquidCrystal_I2C lcd(0x27, 16, 2);
6
7  // Création instance MPU6050
8  MPU6050 mpu;
9  // Variables pour stocker l'accélération en g (gravité)
10 float ax, ay, az;
11 // Norme totale de l'accélération (valeur absolue)
12 float accel_norm;
13 // Seuil d'accélération (en g) pour détecter un mouvement significatif
14 const float threshold = 0.2;
15
16 void setup() {
17   Serial.begin(9600); // Initialisation communication série à 9600 bauds (pour debug éventuel)
18   Wire.begin();       // Initialisation bus I2C
19
20   lcd.init();          // Initialisation écran LCD
21   lcd.backlight();     // Allumer rétroéclairage LCD
22
23   // Initialisation MPU6050
24   mpu.initialize();
25
26   // Vérification de la connexion avec MPU6050
27   if (!mpu.testConnection()) {
28     lcd.print("MPU6050 error"); // Affiche erreur si capteur non détecté
29     while (1);                 // Bloque le programme ici (boucle infinie)
30   }
31 }
```





# Code Arduino

sketch\_jun11a.ino

```
31
32  lcd.clear();
33  lcd.print("MPU6050 OK"); // Confirmation que capteur est prêt
34  delay(1000);
35  lcd.clear();
36 }
37
38 void loop() {
39  // Variables pour stocker les valeurs brutes (raw) lues depuis MPU6050
40  int16_t rawAx, rawAy, rawAz;
41
42  // Lecture des valeurs d'accélération brute sur 3 axes
43  mpu.getAcceleration(&rawAx, &rawAy, &rawAz);
44
45  // Conversion des valeurs brutes en unités g (gravité)
46  // Selon datasheet, la sensibilité est 16384 LSB/g en mode ±2g
47  ax = rawAx / 16384.0;
48  ay = rawAy / 16384.0;
49  az = rawAz / 16384.0;
50
51  // Calcul de la norme de l'accélération totale vectorielle
52  // sqrt(x² + y² + z²) -> donne l'intensité totale ressentie
53  accel_norm = sqrt(ax * ax + ay * ay + az * az);
54
```



# Code Arduino

sketch\_jun11a.ino

```
55 String direction = ""; // Texte à afficher (direction détectée)
56 if (ax > threshold) {
57     direction = "Gauche"; } // Si acceleration sur axe X positive > seuil
58
59 else if (ax < -threshold) {
60     direction = "Droite"; } // Si acceleration sur axe X négative < -seuil
61
62 else if (ay > threshold) {
63     direction = "Arriere"; } // Si acceleration sur axe Y positive > seuil
64
65 else if (ay < -threshold) {
66     direction = "Avant"; } // Si acceleration sur axe Y négative < -seuil
67
68 else if (az > threshold) {
69     direction = "Haut"; } // Si acceleration sur axe Z positive > seuil
70
71 else if (az < -threshold) {
72     direction = "Bas"; } // Si acceleration sur axe Z négative < -seuil
73
74 else {
75     direction = "Stable"; // Si aucune acceleration significative détectée
76 }
77 // Affichage sur l'écran LCD
78 lcd.clear(); // Efface l'écran à chaque boucle
79 lcd.setCursor(0, 0); // Position curseur ligne 0, colonne 0
80 lcd.print("Dir: "); // Affiche label "Direction"
81 lcd.print(direction); // Affiche la direction détectée
82
83 lcd.setCursor(0, 1); // Position curseur ligne 1, colonne 0
84 lcd.print("Accel: "); // Affiche label "Accélération"
85 lcd.print(accel_norm, 2); // Affiche la norme de l'accélération avec 2 décimales
86 lcd.print(" g"); // Affiche unité g (gravité)
```



# Vidéo de démonstration du fonctionnement du capteur MPU6050



# Merci pour votre attention