

# Rapport du Projet

## 4DOF Mechanical robot arm car

### PRÉSENTÉ PAR:

NASSIF Walid  
OUCHKER MOHAMED AYMAN  
HASSNA AIT OURAHIM

### ENCADRÉ PAR:

MR.CHALH Zakaria

03/01/2024



# I.Introduction :

Les bras mécaniques, reproduisant la flexibilité des bras humains, sont capables d'adopter diverses positions. Actuellement, de nombreux bras mécaniques se trouvent limités dans leur flexibilité en raison des variations d'environnement et de distance.

Face à ce défi, le groupe KEYES a développé un kit d'apprentissage 3 en 1 : un bras robotique mécanique **4DOF** intégré à un robot mobile. Ce qui offre l'opportunité d'acquérir des compétences de contrôle tant pour un bras mécanique que pour une voiture intelligente.

## II.Caractéristiques :

**Design 3 en 1** : Voiture intelligente, bras mécanique, robot mobile à bras mécanique, fonction polyvalente : évitement d'obstacles, suivi, télécommande et convoyage automatique.

**Facile à construire** : Aucune soudure de circuit n'est nécessaire.

**Haute ténacité** : Plaque de base haute performance et bras mécanique en métal

**Haute extension** : Ajoutez d'autres capteurs et modules via le module de commande moteur.






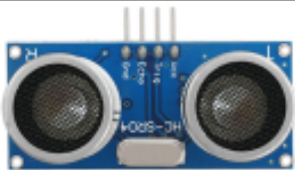

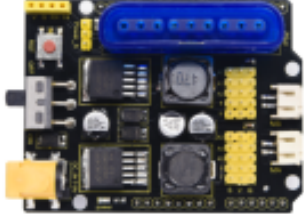

**Contrôles multiples** : Commande par manette PS2, entièrement automatique et contrôle via une application (systèmes iOS et Android).

**Programmation de base** : Apprentissage du langage C. Kit d'apprentissage pour bras mécanique 4DOF de Keyestudio avec voiture robotique et manette PS2 pour Arduino, compatible avec les systèmes Android et iOS.

### III.Spécifications :

- Tension de fonctionnement : 5V
- Tension d'entrée : 7-12V
- Courant de sortie maximal : 3A
- Puissance maximale dissipée : 25W (T=75°C)
- Vitesse du moteur : 5V 63 tours/minute
- Forme de commande du moteur : TB6612
- Angle de détection ultrasonique : <15 degrés
- Distance de détection ultrasonique : 2 cm - 400 cm
- Distance de contrôle à distance Bluetooth : 20-50 mètres (mesurée)
- Contrôle Bluetooth via application : prend en charge les systèmes Android et iOS

### IV. Description matériel :

Model	QTY	Picture	KEYESTUDIO Red LED Module	1	
KEYESTUDIO V4.0 Development Board (Compatible <u>Arduino UNO</u> )	1		PS2 Wireless 2.4G Game Controller	1	
USB Cable AM/BM Blue OD:5.0 L=1m	1		KEYESTUDIO HM-10 Bluetooth-4.0 V3 Module	1	
HC-SR04 Ultrasonic Sensor	1		MG90S 14G Servo	4	
KEYESTUDIO TB6612FNG Motor/Servo Drive Shield	1		Bearing Cap	1	

# V.Objectifs :

## 1.Assemblage du Robot :

- Comprendre les pièces du kit 4DOF.
- Assemblage de la base, du bras mécanique, et des roues en suivant les instructions fournies.

## 2.Cinématique Directe :

- Explorer les principes de base de la cinématique directe appliquée au bras mécanique 4DOF.
- Comprendre comment les mouvements des joints contribuent à la position et à l'orientation du bras.

## 3.Programmation du Bras Mécanique :

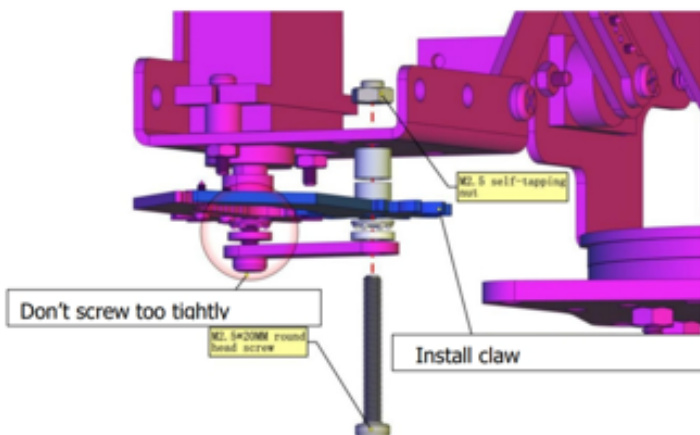
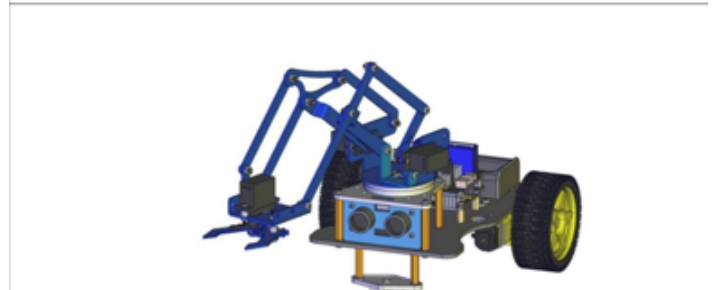
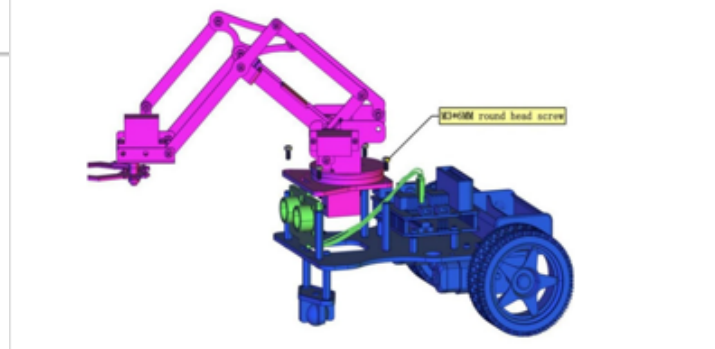
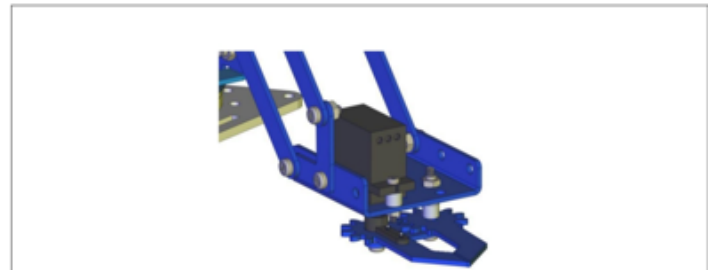
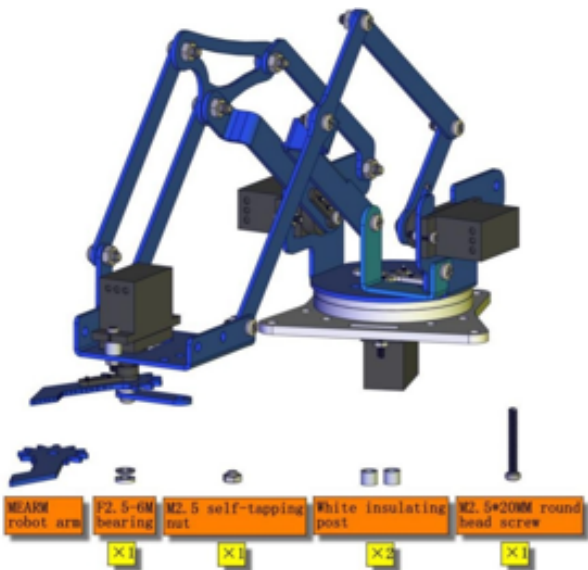
- Écrire des programmes pour contrôler les mouvements du bras mécanique en tenant compte des angles des articulations.
- Expérimenter avec des séquences de mouvements programmées pour explorer les capacités du bras mécanique.

# 1-Assemblage du Robot :

## i-Le kit 4DOF:

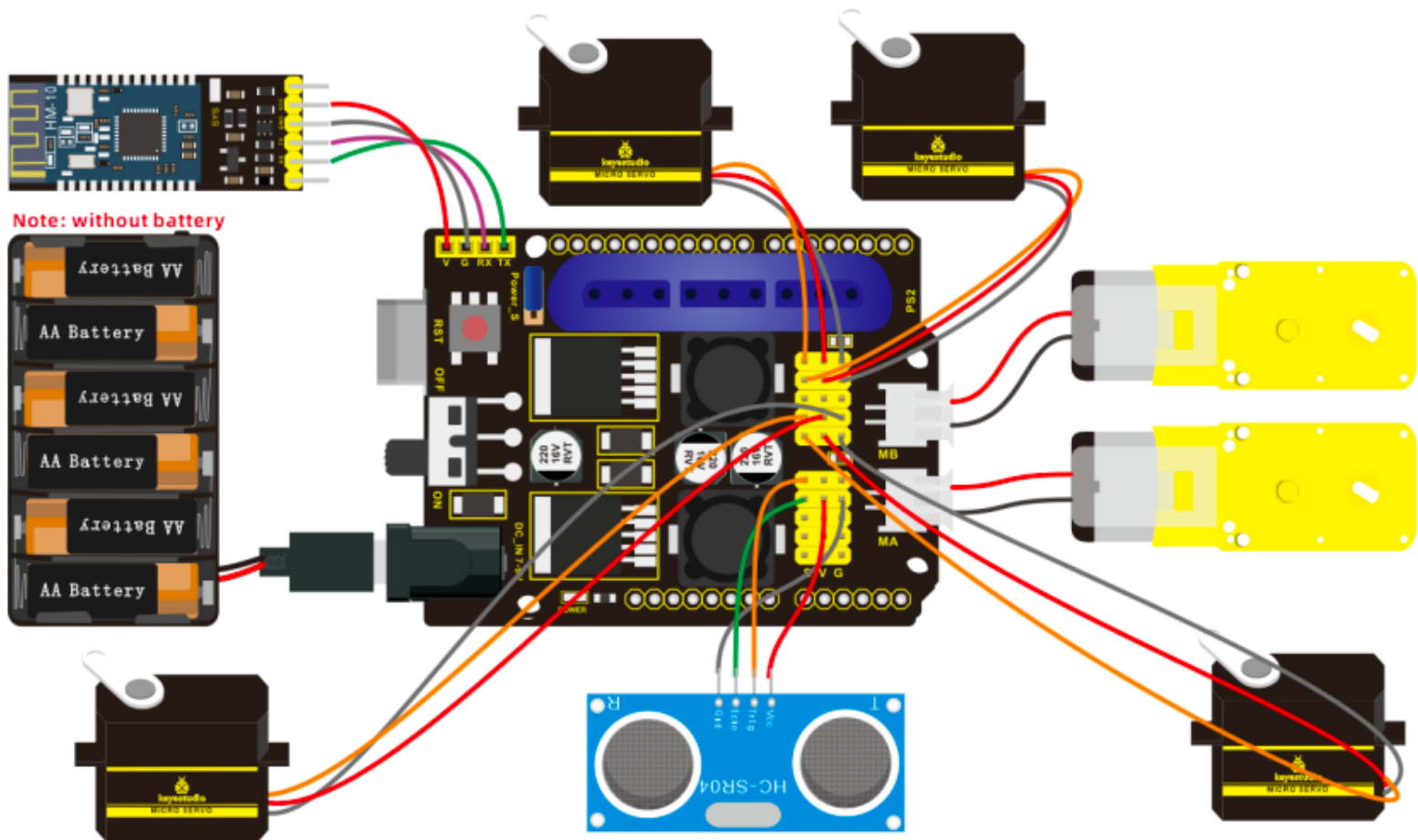


## ii-Assemblage:





### iii-schéma de connexion:

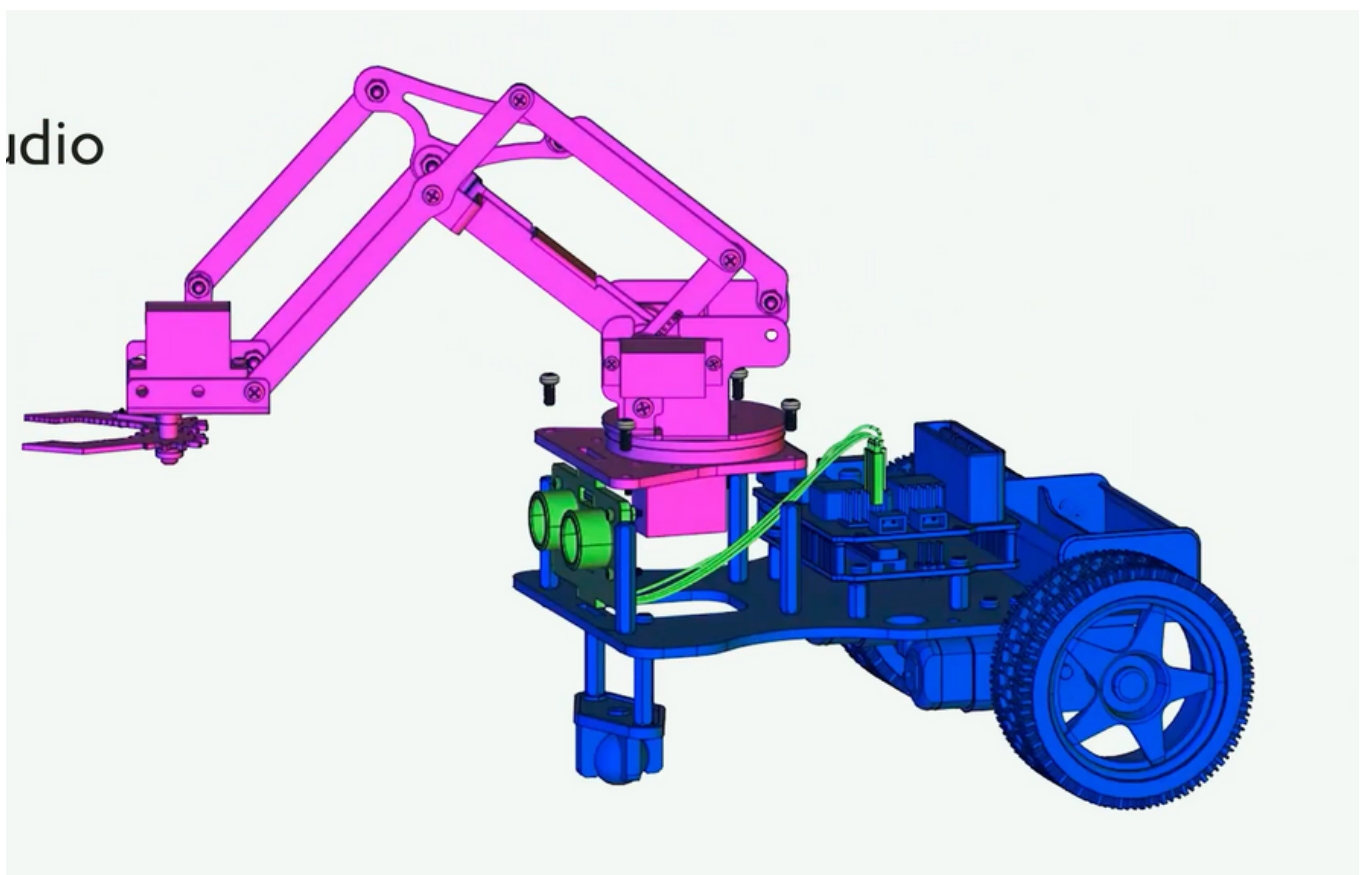


## 2-Cinématique directe :

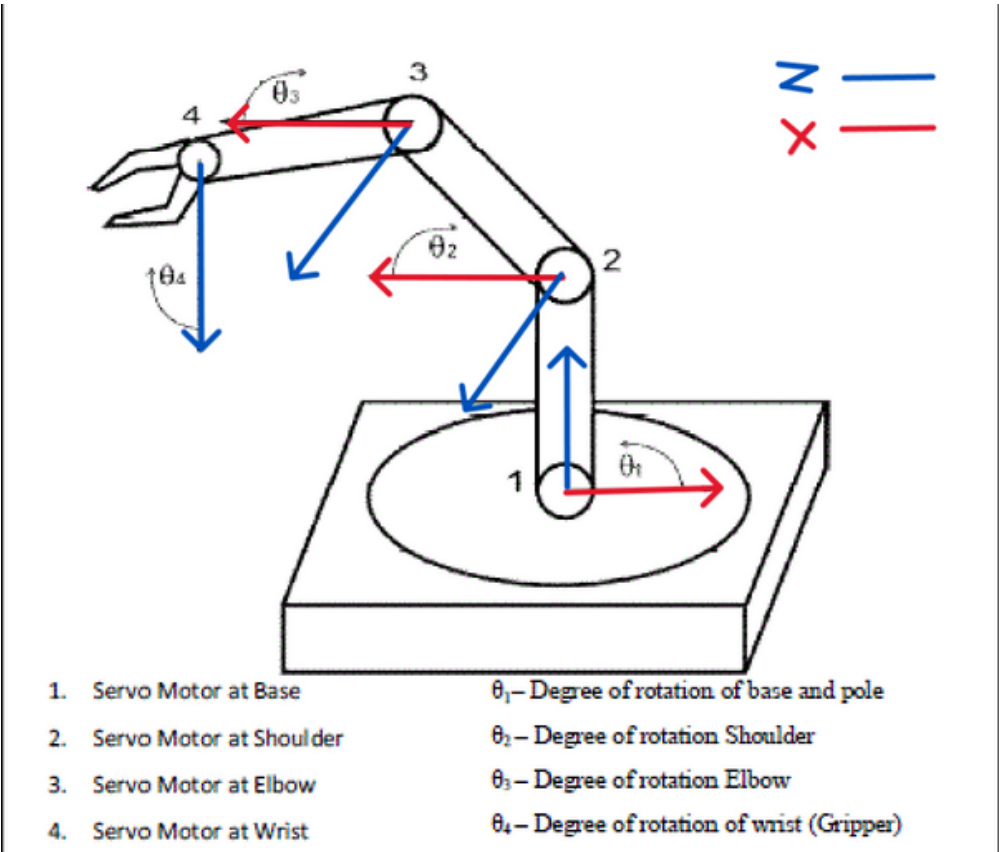
### i-Modele 3D :

Le bras mécanique, intégré à un robot mobile à roues et contrôlé par une manette, possède 4 Degrés de Liberté (DOF). Ces degrés de liberté permettent au bras mécanique d'effectuer des mouvements dans des directions ou axes indépendants, facilitant différentes fonctionnalités dans le contexte d'une application mobile.

1. **Rotation** : Le bras peut tourner sur lui-même autour de son axe vertical, permettant une orientation variable du dispositif de préhension.
2. **Extension** : Le bras peut s'étendre ou se rétracter, réalisant une translation le long de son axe horizontal, adaptant ainsi la portée du bras.
3. **Montée/Descente** : Le bras peut monter et descendre le long d'un axe vertical, ajustant la hauteur du dispositif de préhension par rapport au sol.
4. **Préhension** : Le bras peut réaliser un mouvement de pincement, ouvrir et fermer une pince, facilitant la manipulation d'objets.



ii- Referentiel DH :



iii- Paramètres DH :

i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	$d_1(1)$	0	$0^\circ$
2	$\theta_2$	$d_2(2)$	0	$-90^\circ$
3	$\theta_3+90$	0	$a_3(7)$	$0^\circ$
4	$\theta_4+90$	0	$a_4(10)$	$-90^\circ$



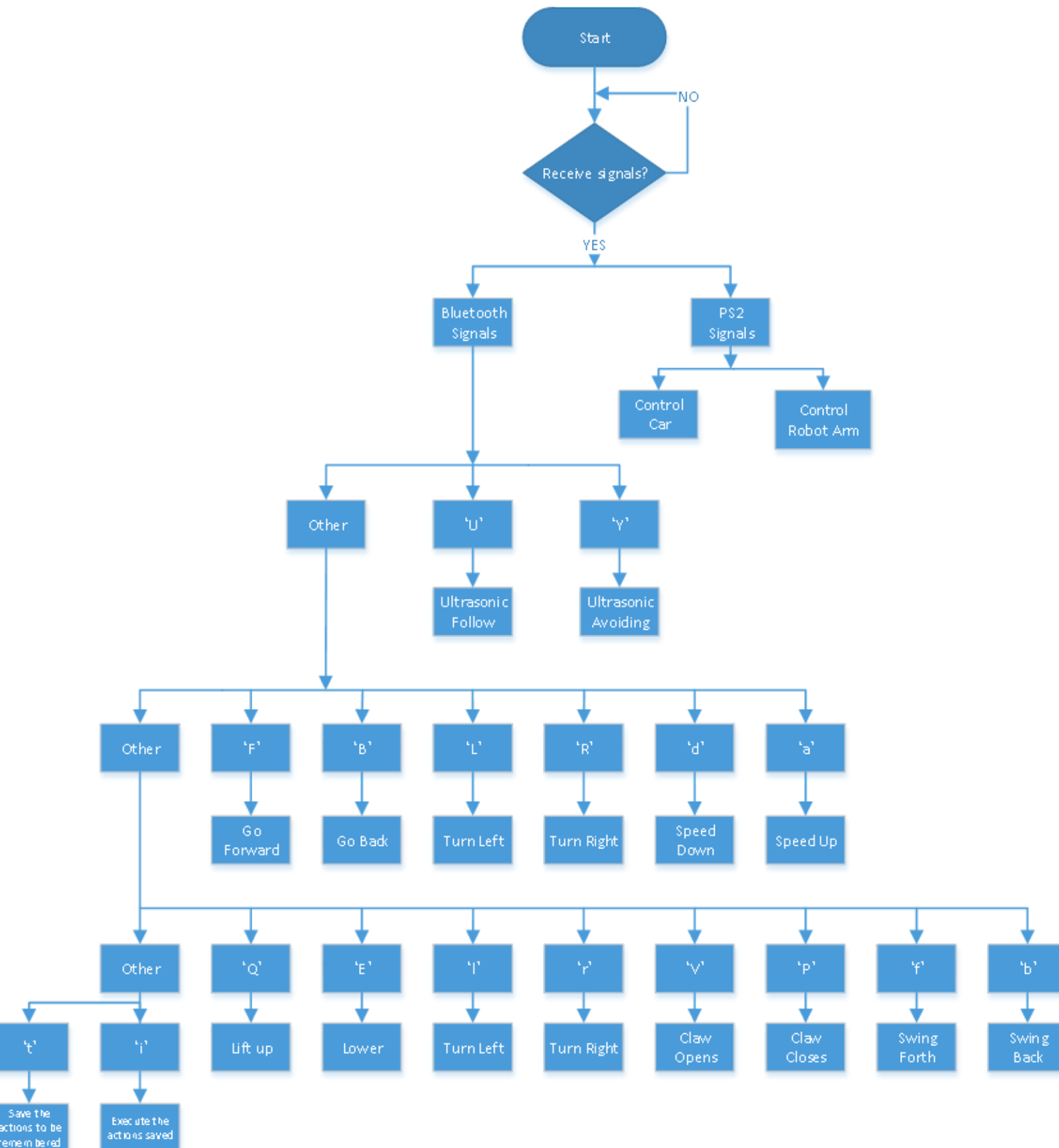
$H_{i-1/i} = H_{rot,z}(\theta_1) * H_{trans}(0, 0, d_i) * H_{trans}(a_i, 0, 0) * H_{rot,x}(\alpha_1):$

H0/4=

0.97	-0.174	0.17	8.58
0.17	0.98	0.03	1.005
-0.174	0	0.98	-5.6
0	0	0	1

### 3- Programmation :

#### i-Organigramme :



# ii-code:



sketch\_jan4a.ino

```
1 #include <PS2X_lib.h> //add the library of ps2 handle
2 #include <Servo.h> //add the library of servo
3 PS2X ps2x; // create PS2 Controller Class
4 #define PS2_DAT 12 //ps2receiver
5 #define PS2_CMD 11
6 #define PS2_SEL 10
7 #define PS2_CLK 13
8 Servo myservo1; //define the name of servo variable
9 Servo myservo2; //define the name of servo variable
10 Servo myservo3; //define the name of servo variable
11 Servo myservo4; //define the name of servo variable
12 int error = 0;
13 byte vibrate = 0;
14 int pos1 = 90, pos2 = 100, pos3 = 80, pos4 = 90; // define angle variable of four servos and set initial value(posture angle value when setting up)
15 char blue_val;
16 int M1[20]; //define four arrays
17 int M2[20]; //respectively save the angle of four servos
18 int M3[20]; //the array length is 20, can save 0~20 angle data
19 int M4[20];
20 int i = 0, j = 0, t = 0; //i is used to save array, j is used to save the maximum value of i,t is used to exit while loop
21 const int AIN2 = 2; //define the driving pins of motor
22 const int PWM_A = 3;
23 const int BIN2 = 4; //when AIN2 is low and AIN1 is high, BIN2 is high and BIN1 is low
24 const int PWM_B = 5;
25 int echoPin = A3; // ultrasonic module ECHO to A3
26 int trigPin = A4; // ultrasonic module TRIG to A4
27 int speeds = 100; // set the initial value of rotation speed of motor
28 int Ultrasonic_Ranging() { //ultrasonic ranging function
29     digitalWrite(trigPin, LOW);
30     delayMicroseconds(2);
31     digitalWrite(trigPin, HIGH);
32     delayMicroseconds(10); //send least 10us high level to trig pin to trigger ultrasonic waves
33     digitalWrite(trigPin, LOW);
34     int distance = pulseIn(echoPin, HIGH); // reading the duration of high level
35     distance = distance / 58; // Transform pulse time to distance
36     delay(30);
37     return distance; //return distance to this function
38 }
39 void setup() {
40     Serial.begin(9600); //set to baud rate to 57600 when printing ps2, however, Bluetooth can't be used
41     myservo1.attach(A1); //set control pin of servo 1 to A1
42     myservo2.attach(A0); //set control pin of servo 2 to A0
43     myservo3.attach(8); //set control pin of servo 3 to D8
44     myservo4.attach(9); //set control pin of servo 4 to D9
45     myservo3.write(pos3); //servo 3 rotates to 80°
46     delay(500);
47     myservo2.write(pos2); //servo 2 rotates to 100°
48     delay(500);
49     myservo1.write(pos1); //servo 1 rotates to 90°
50     delay(500);
51     myservo4.write(pos4); //servo 4 rotates to 90°
52     pinMode(2, OUTPUT); //set ports of motor to output
53     pinMode(3, OUTPUT);
54     pinMode(4, OUTPUT);
55     pinMode(5, OUTPUT);
56     pinMode(echoPin, INPUT); //set echoPin to input
57     pinMode(trigPin, OUTPUT); //set trigPin to output
58     error = ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_SEL, PS2_DAT); //setup pins check for error
59     if (error == 0) {
60         Serial.println("Found Controller, configured successful");
61         Serial.println("Try out all the buttons, X will vibrate the controller, faster as you press harder;");
62         Serial.println("holding L1 or R1 will print out the analog stick values.");
63         Serial.println("Go to www.billporter.info for updates and to report bugs.");
64     }
65     else if (error == 1) Serial.println("No controller found, check wiring, see readme.txt to enable debug. visit www.billporter.info for troubleshooting tips");
66     else if (error == 2) Serial.println("Controller found but not accepting commands. see readme.txt to enable debug. Visit www.billporter.info for troubleshooting tips");
67     ps2x.enableRumble(); //enable rumble vibration motors
68     ps2x.enablePressures(); //enable reading the pressure values from the buttons.
69 }
70 }
71 void loop() {
```