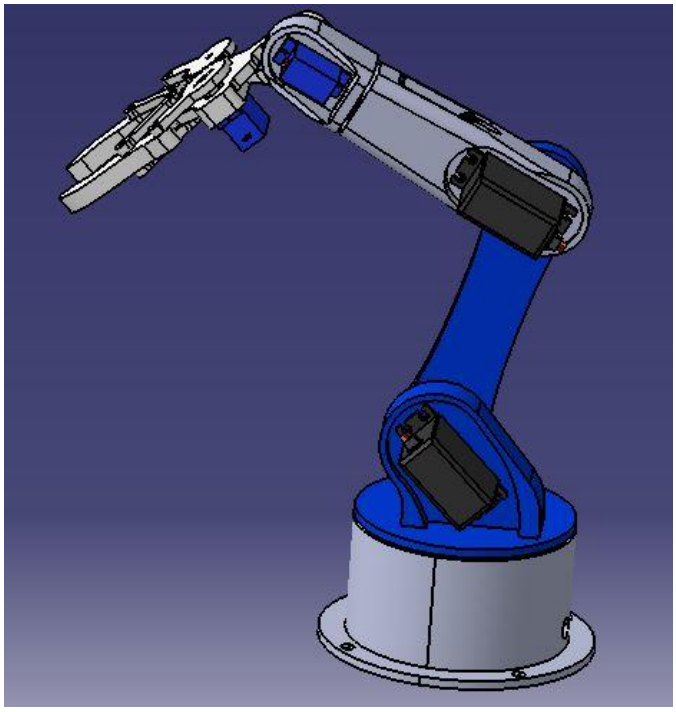
# Bras industriel autonome et contrôler par application Android

## Introduction :

## Conception du bras sur catia

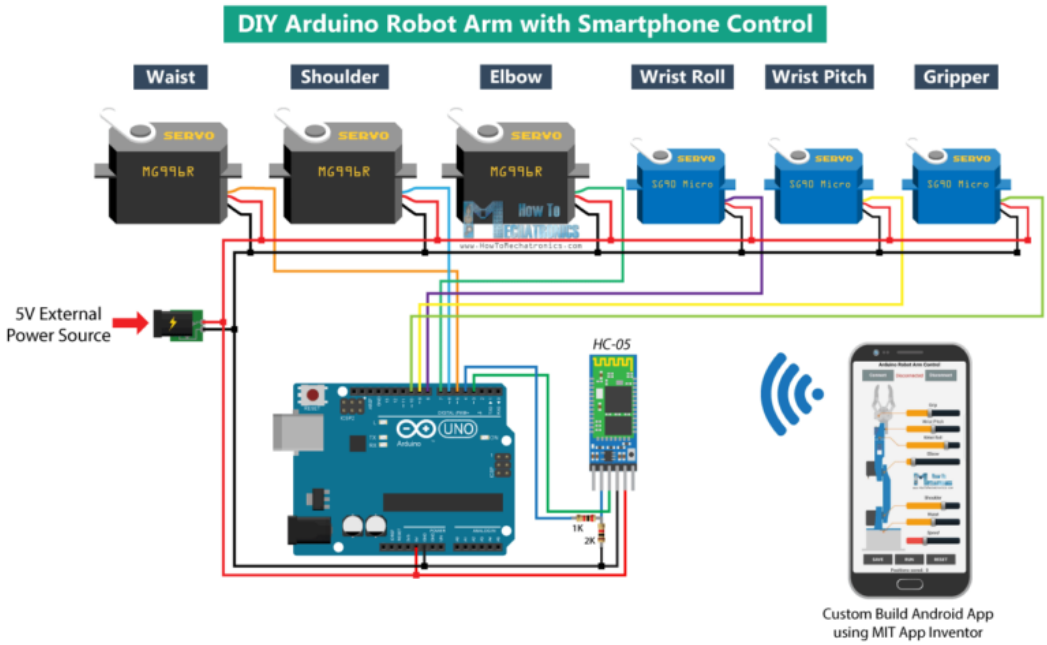
Pour les 3 premiers axes, la taille, l'épaule et le coude, j'ai utilisé les servos MG996R, et pour les 2 autres axes, le roulis et le pas du poignet, ainsi que la pince, j'ai utilisé les micro-servos SG90 plus petits.

Vous pouvez télécharger et le modèle 3D ci-dessous.



## Diagramme du circuit du bras du robot Arduino

L'étape suivante consiste à connecter l'électronique. Le schéma de circuit de ce projet est en fait assez simple. Nous avons juste besoin d'une carte Arduino et d'un module Bluetooth HC-05 pour communiquer avec le smartphone. Les broches de contrôle des six servomoteurs sont connectées à six broches numériques de la carte Arduino.




## Composants matériels

1. **MG996R Servo Motor**


2. **SG90 Micro Servo Motor**


3. **HC-05 Bluetooth Module**

4. **Arduino Board**


5. **5V 2A DC Power Supply**

## Code du bras du robot Arduino

Nous devons d'abord inclure la bibliothèque SoftwareSerial pour la communication série du module Bluetooth ainsi que la bibliothèque d'asservissement. Ces deux bibliothèques sont incluses avec l'IDE Arduino afin que vous n'ayez pas à les installer en externe. Ensuite, nous devons définir les six servos, le module Bluetooth HC-05 et certaines variables pour stocker la position actuelle et précédente des servos, ainsi que des tableaux pour stocker les positions ou les étapes pour le mode automatique.

```
#include <SoftwareSerial.h>
#include <Servo.h>

Servo servo01;
Servo servo02;
Servo servo03;
Servo servo04;
Servo servo05;
Servo servo06;

SoftwareSerial Bluetooth(3, 4); // Arduino(RX, TX) - HC-05 Bluetooth (TX, RX)

int servo1Pos, servo2Pos, servo3Pos, servo4Pos, servo5Pos, servo6Pos; // current position
int servo1PPos, servo2PPos, servo3PPos, servo4PPos, servo5PPos, servo6PPos; // previous position
int servo01SP[50], servo02SP[50], servo03SP[50], servo04SP[50], servo05SP[50], servo06SP[50]; // for storing positions/steps
int speedDelay = 20;
int index = 0;
String dataIn = "";
```

Dans la section de configuration, nous devons initialiser les servos et le module Bluetooth et déplacer le bras du robot dans sa position initiale. Nous faisons cela en utilisant la fonction write () qui déplace simplement le servo à n'importe quelle position de 0 à 180 degrés.

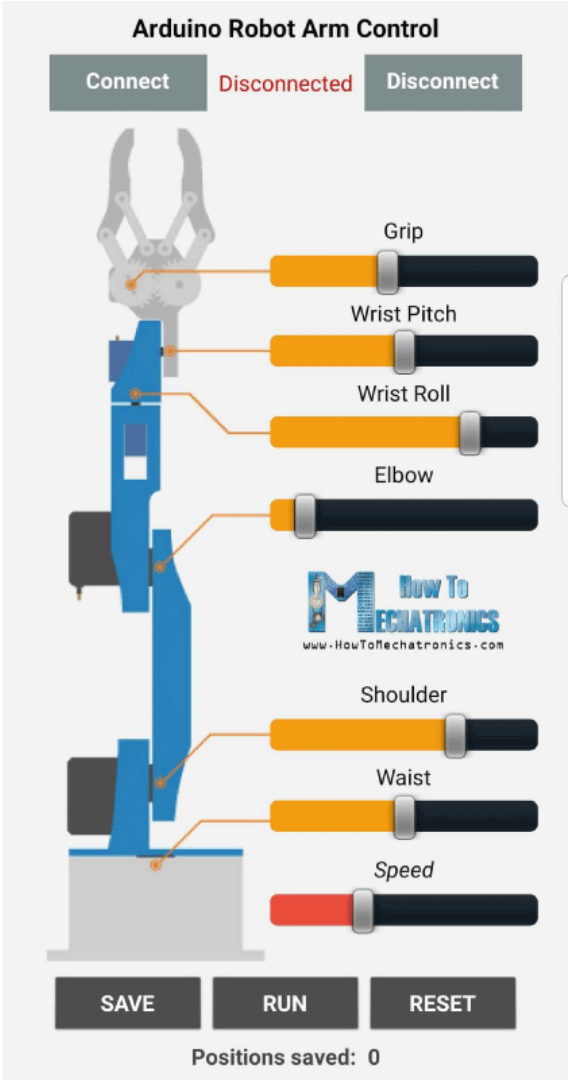
```
void setup() {
  servo01.attach(5);
  servo02.attach(6);
  servo03.attach(7);
  servo04.attach(8);
  servo05.attach(9);
  servo06.attach(10);
  Bluetooth.begin(38400); // Default baud rate of the Bluetooth module
  Bluetooth.setTimeout(1);
  delay(20);
  // Robot arm initial position
  servo1PPos = 90;
  servo01.write(servo1PPos);
  servo2PPos = 150;
  servo02.write(servo2PPos);
  servo3PPos = 35;
  servo03.write(servo3PPos);
  servo4PPos = 140;
  servo04.write(servo4PPos);
  servo5PPos = 85;
  servo05.write(servo5PPos);
  servo6PPos = 80;
  servo06.write(servo6PPos);
}
```

Ensuite, dans la section boucle, en utilisant la fonction Bluetooth.available (), nous vérifions constamment s'il y a des données entrantes du Smartphone. Si vrai, en utilisant la fonction readString (), nous lisons les données sous forme de chaîne et les stockons dans la variable dataIn. En fonction des données arrivées, nous dirons au bras du robot quoi faire.

```
// Check for incoming data
  if (Bluetooth.available() > 0) {
    dataIn = Bluetooth.readString();  // Read the data as string
```

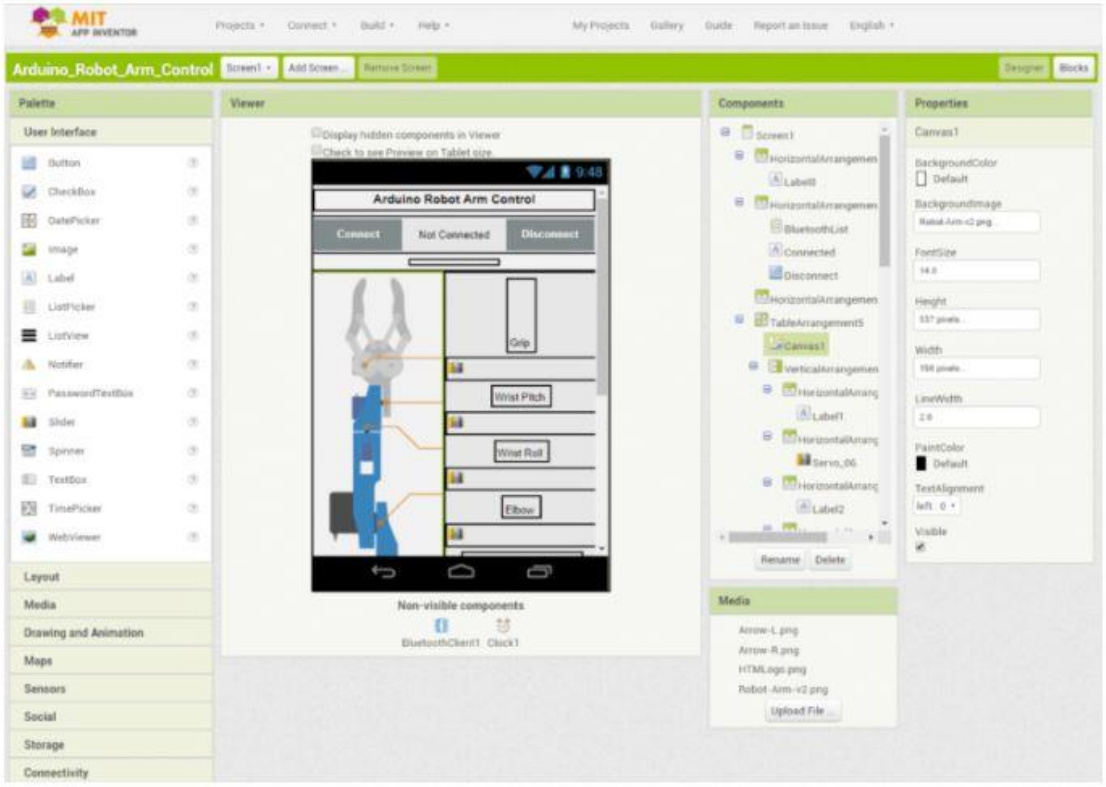
## Application Android Arduino Robot Arm Control

Jetons un coup d'œil à l'application Android maintenant et voyons quel type de données elle envoie réellement à l'Arduino.



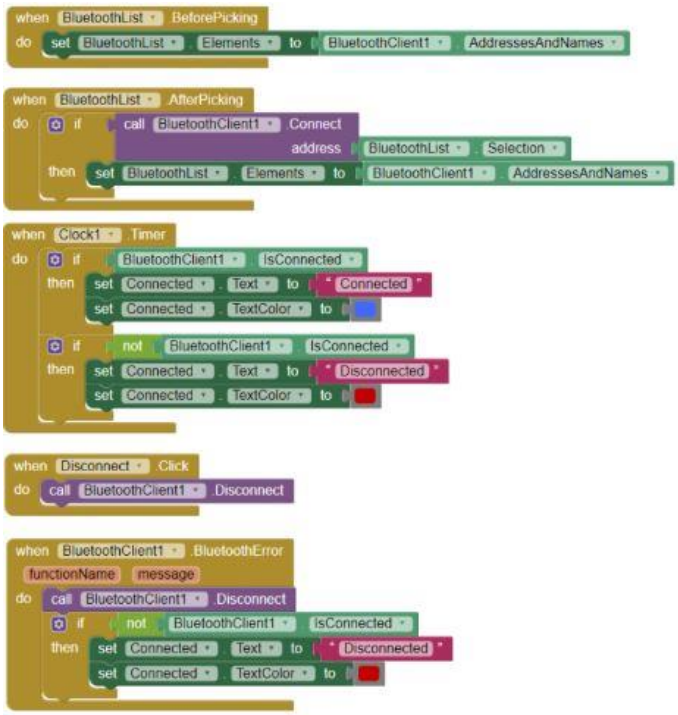

I made the app using the MIT App Inventor online application and here's how it works. At the top we have two buttons for connecting the smartphone to the HC-05 Bluetooth module. Then on the left side we have

an image of the robot arm, and on the right side we have the six sliders for controlling the servos and one slider for the speed control.

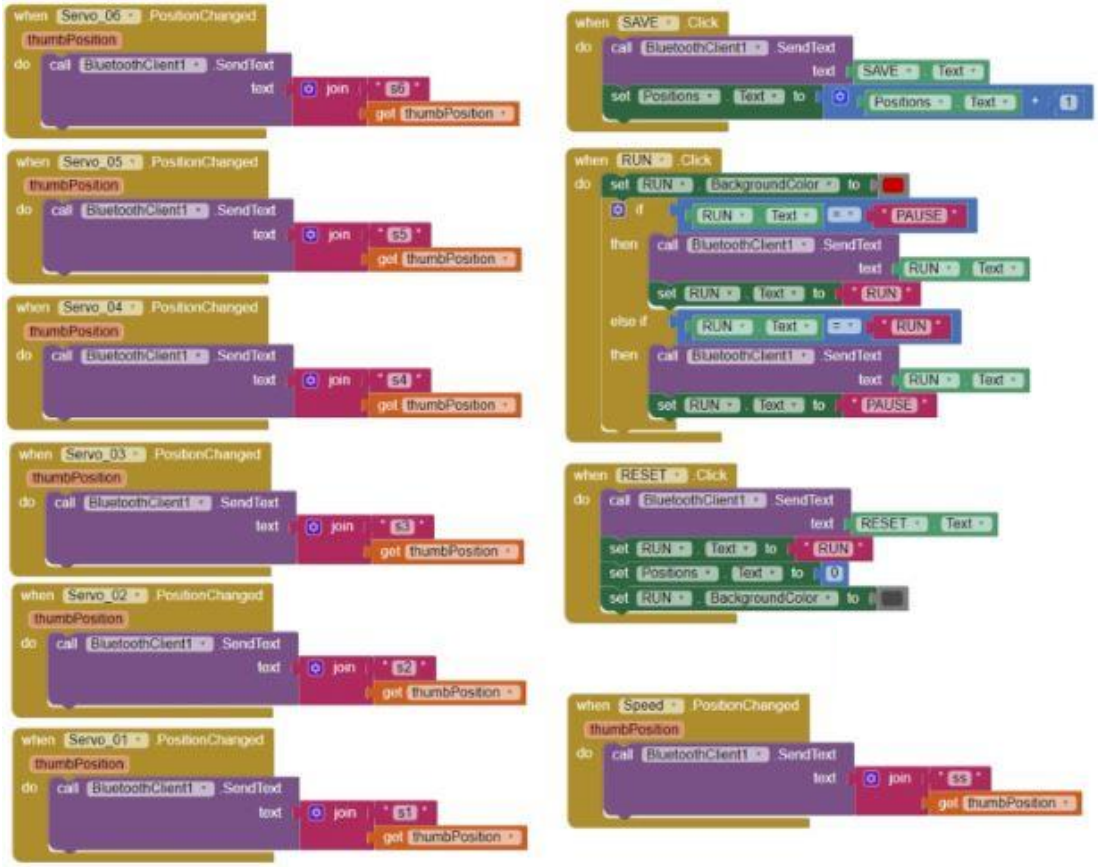



Chaque curseur a une valeur initiale, minimale et maximale différente qui convient aux articulations du bras du robot. Au bas de l'application, nous avons trois boutons, SAVE, RUN et RESET à travers lesquels nous pouvons programmer le bras du robot pour qu'il s'exécute automatiquement. Il y a aussi une étiquette ci-dessous qui montre le nombre d'étapes que nous avons enregistrées.

Ok, voyons maintenant le programme ou les blocs derrière l'application. Tout d'abord, sur le côté gauche, nous avons les blocs pour connecter le smartphone au module Bluetooth.




Ensuite, nous avons les blocs curseurs pour le contrôle de la position du servo et les blocs boutons pour programmer le bras du robot. Donc, si nous changeons la position du curseur, en utilisant la fonction Bluetooth. SendText, nous envoyons un texte à l'Arduino. Ce texte se compose d'un préfixe qui indique quel curseur a été modifié ainsi que la valeur actuelle du curseur.




Here’s a download file of the above MIT App Inventor project, as well as the Android App ready to be installed on your smartphone:

Arduino Robot Arm Control MIT App Inventor Project File

https://howtomechatronics.com/tutorials/arduino/diy-arduino-robot-arm-with-smartphone-control/#

Arduino Robot Arm Control Android App

https://howtomechatronics.com/tutorials/arduino/diy-arduino-robot-arm-with-smartphone-control/#

Donc, à l'Arduino, en utilisant la fonction startsWith () nous vérifions le préfixe de chaque donnée entrante et nous savons donc quoi faire ensuite. Par exemple, si le préfixe est «s1», nous savons que nous devons déplacer le servo numéro un. En utilisant la fonction substring (), nous obtenons le texte restant, ou c'est la valeur de position, nous le convertissons en entier et utilisons la valeur pour déplacer le servo à cette position.

```
// If "Waist" slider has changed value - Move Servo 1 to position
    if (dataIn.startsWith("s1")) {
      String dataInS = dataIn.substring(2, dataIn.length()); // Extract only the number. E.g. from "s1120" to "120"
      servo1Pos = dataInS.toInt();  // Convert the string into integer
```

Ici, nous pouvons simplement appeler la fonction write () et le servo ira à cette position, mais de cette façon, le servo fonctionnerait à sa vitesse maximale, ce qui est trop pour le bras du robot. Au lieu de cela, nous devons contrôler la vitesse des servos, j'ai donc utilisé des boucles FOR afin de déplacer progressivement le servo de la position précédente à la position actuelle en implémentant un temps de retard entre chaque itération. En modifiant le temps de retard, vous pouvez modifier la vitesse du servo.

```
// We use for loops so we can control the speed of the servo
      // If previous position is bigger then current position
      if (servo1PPos > servo1Pos) {
        for ( int j = servo1PPos; j >= servo1Pos; j--) {   // Run servo down
          servo01.write(j);
          delay(20);    // defines the speed at which the servo rotates
        }
      }
      // If previous position is smaller then current position
      if (servo1PPos < servo1Pos) {
        for ( int j = servo1PPos; j <= servo1Pos; j++) {   // Run servo up
          servo01.write(j);
          delay(20);
        }
      }
      servo1PPos = servo1Pos;   // set current position as previous position
    }
```

.La même méthode est utilisée pour entraîner chaque axe du bras du robot

En dessous d'eux se trouve le bouton SAVE. Si nous appuyons sur le bouton SAVE, la position de chaque servomoteur est stockée dans un tableau. À chaque pression, l'index augmente de sorte que le tableau est rempli pas à pas.

```
// If button "SAVE" is pressed
    if (dataIn.startsWith("SAVE")) {
      servo01SP[index] = servo1PPos;  // save position into the array
      servo02SP[index] = servo2PPos;
      servo03SP[index] = servo3PPos;
      servo04SP[index] = servo4PPos;
      servo05SP[index] = servo5PPos;
      servo06SP[index] = servo6PPos;
      index++;                        // Increase the array index
    }
```

Ensuite, si nous appuyons sur le bouton RUN, nous appelons la fonction personnalisée runservo () qui exécute les étapes stockées. Jetons un œil à cette fonction. Ici, nous exécutons les étapes stockées encore et encore jusqu'à ce que nous appuyions sur le bouton RESET. En utilisant la boucle FOR, nous parcourons toutes les positions stockées dans les tableaux et en même temps, nous vérifions si nous avons des données entrantes du smartphone. Ces données peuvent être le bouton RUN / PAUSE, qui interrompt le robot et, si vous cliquez à nouveau, continue avec les mouvements automatiques. De plus, si nous modifions la position du curseur de vitesse, nous utiliserons cette valeur pour modifier le temps de retard entre chaque itération dans les boucles FOR ci-dessous, qui contrôle la vitesse des servomoteurs.

```
// Automatic mode custom function - run the saved steps
void runservo() {
  while (dataIn != "RESET") {   // Run the steps over and over again until "RESET" button is pressed
    for (int i = 0; i <= index - 2; i++) {  // Run through all steps(index)
      if (Bluetooth.available() > 0) {      // Check for incomding data
        dataIn = Bluetooth.readString();
        if ( dataIn == "PAUSE") {           // If button "PAUSE" is pressed
          while (dataIn != "RUN") {         // Wait until "RUN" is pressed again
            if (Bluetooth.available() > 0) {
              dataIn = Bluetooth.readString();
              if ( dataIn == "RESET") {
                break;
              }
            }
          }
        }
        // If SPEED slider is changed
        if (dataIn.startsWith("ss")) {
          String dataIn8 = dataIn.substring(2, dataIn.length());
          speedDelay = dataIn8.toInt(); // Change servo speed (delay time)
        }
      }
      // Servo 1
      if (servo018P[i] == servo018P[i + 1]) {
      }
      if (servo018P[i] > servo018P[i + 1]) {
        for ( int j = servo018P[i]; j >= servo018P[i + 1]; j--) {
          servo01.write(j);
          delay(speedDelay);
        }
      }
      if (servo018P[i] < servo018P[i + 1]) {
        for ( int j = servo018P[i]; j <= servo018P[i + 1]; j++) {
          servo01.write(j);
          delay(speedDelay);
        }
      }
```

De la même manière qu'expliqué précédemment avec ces instructions IF et boucles FOR, nous déplaçons les servos à leur position suivante. Enfin, si nous appuyez sur le bouton RESET, nous effacerons toutes les données des tableaux à zéro et remettrons également l'index à zéro afin de pouvoir reprogrammer le bras du robot avec de nouveaux mouvements.

```
// If button "RESET" is pressed
    if ( dataIn == "RESET") {
      memset(servo01SP, 0, sizeof(servo01SP)); // Clear the array data to 0
      memset(servo02SP, 0, sizeof(servo02SP));
      memset(servo03SP, 0, sizeof(servo03SP));
      memset(servo04SP, 0, sizeof(servo04SP));
      memset(servo05SP, 0, sizeof(servo05SP));
      memset(servo06SP, 0, sizeof(servo06SP));
      index = 0;  // Index to 0
    }
```