THE NANOBOT

Our project is called the nanobot. It is a robotic arm which can be controlled remotely from a device. Our device is a compact rotatable device made of 3d printed body which can lift about 100-200 grams of weight. Our main aim was to maneuver the device seamlessly and pick and drop the weights with precision.

COMPONENTS

ESP-32
Used for the communication between the Gripper and the device.

Servo Motors
The servo enables movement in the device

Battery
Two 3.7-volt lithium-ion batteries have been used to power the device.

5V Voltage Regulator
Protects the gadget against high voltage.

Body Structure
3D-printed precision acrylic parts have been used for the body structure.

Male to Female Jumper Wires
Used to make connections between the servo and the module

Switch
A simple red switch for turning our device on and off

THE CODE

The Nanobots code enables control of the gripper using ESP32. It receives commands from the device enabling movements across three servo motors: base, elbow, claw. The "smoothMove” function ensures precise motion for each servo motor which enhances our accuracy. A “reset’ function brings the arm back to its default position.

```
#include <ESP32Servo.h>
#include <BluetoothSerial.h>

BluetoothSerial SerialBT;

Servo base;
Servo el;
Servo claw;

// Track current positions
int baseAngle = 90;
int elAngle = 90;
int clawAngle = 90;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("Robotic_Arm");
  Serial.println(" Bluetooth Ready. Waiting for commands...");

  base.attach(15);
  el.attach(17);
  claw.attach(19);

  // Set all to starting positions
  base.write(baseAngle);
  el.write(elAngle);
  claw.write(clawAngle);}
//  Smooth movement function
void smoothMove(Servo &servo, int &currentAngle, int targetAngle, int delayMs = 5) {
  if (currentAngle == targetAngle) return;
  int step = (currentAngle < targetAngle) ? 1 : -1;

  for (int pos = currentAngle; pos != targetAngle; pos += step) {
    servo.write(pos);
    delay(delayMs);
  }
  servo.write(targetAngle); // Ensure final position
  currentAngle = targetAngle;
}
void loop() {
  if (SerialBT.available()) {
    char command = SerialBT.read();
    Serial.print("Received: ");
    Serial.println(command);

    // --- Base rotation ---
    if (command == '1') {
     smoothMove(base, baseAngle, 30);
     Serial.println("Base: Left");
    } else if (command == '2') {
     smoothMove(base, baseAngle, 150);
     Serial.println("Base: Right");
    } else if (command == '0') {
     smoothMove(base, baseAngle, 90);
     Serial.println("Base: Stop");
    }
     // --- Elbow control ---
    else if (command == '3') {
     smoothMove(el, elAngle, 30);
     Serial.println("Elbow: Down");
    } else if (command == '4') {
     smoothMove(el, elAngle, 130);
     Serial.println("Elbow: Up");



  } else if (command == '7') {
     smoothMove(el, elAngle, 90);
     Serial.println("Elbow: Center");
    }
     // --- Claw control ---
    else if (command == '5') {
     smoothMove(claw, clawAngle, 180);
     Serial.println("Claw: Close");
    } else if (command == '6') {
     smoothMove(claw, clawAngle, 90);
     Serial.println("Claw: Open");
    } else if (command == '8') {
     smoothMove(claw, clawAngle, 135);
     Serial.println("Claw: Center");
    }

    // --- Reset all servos ---
    else if (command == 'r' || command == 'R') {
     resetAll();
    }

    else {
     Serial.println(" Invalid command");
    }
   }
  }

  //  Reset all servos smoothly
  void resetAll() {
   Serial.println(" Resetting all servos...");
   smoothMove(base, baseAngle, 90);
   smoothMove(el, elAngle, 90);
   smoothMove(claw, clawAngle, 180);
   Serial.println(" All Servos Reset to Center (90°)");
   SerialBT.println(" Arm Reset to Default Positions");}
```