Apéndice A. Código de control

sketch_sep22a.ino

```
#include <BluetoothSerial.h>
#include <ESP32Servo.h>
#include "esp_system.h" // Para usar esp_restart()

BluetoothSerial SerialBT;

const int maxSize = 10; // Tamaño máximo del vector
int p[maxSize];         // Vector para almacenar enteros
String values[maxSize]; // Vector para almacenar los strings
int k = 0;              // Índice para el vector

// Pines para los servomotores
const int servoPin1 = 13; // Pin del primer servomotor
const int servoPin2 = 14; // Pin del segundo servomotor

int aiC = 90; // Ángulo inicial Codo
int aiM = 90; // Ángulo inicial Muñeca

volatile bool running = false; // Variable para controlar la ejecución
TaskHandle_t taskHandle = NULL; // Manejo de la tarea

Servo servo1;
Servo servo2;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32_Bluetooth"); // Nombre del dispositivo Bluetooth
  Serial.println("El dispositivo está listo para emparejarse.");

  pinMode(2, OUTPUT); // Configurar el pin 2 como salida

  // Inicializar los servomotores
  servo1.attach(servoPin1);
  servo2.attach(servoPin2);


//  pinMode(15, INPUT);//Potenciometro Codo
//  pinMode(2, INPUT);//Potenciometro Muñeca
//
//  aiC=digitalRead(2)*180/4095;
//  aiM=digitalRead(15)*180/4095;

  // Asegurarse de que los servos están en la posición inicial
  servo1.write(aiC);
  servo2.write(aiM);

}

```

```
void loop() {
  if (SerialBT.available()) {
    String received = SerialBT.readStringUntil('\n'); // Leer hasta el salto de línea

    if (k < maxSize) {
      if (k < 6) { // Guardar los primeros 6 datos como enteros
        int value = received.toInt(); // Convertir a entero
        p[k] = value; // Guardar enteros
        values[k] = String(value); // Almacenar el número como string
        Serial.print("Número recibido: ");
        Serial.println(value);
      } else { // Después de recibir los 6 primeros datos
        Serial.print("String recibido (no almacenado): ");
        Serial.println(received); // Imprimir el string sin almacenarlo

        // Verificar si el string es "Iniciar"
        if (received == "Iniciar") {
          Serial.println("Se ha recibido el comando para iniciar.");
          running = true; // Activar la ejecución

          // Crear la tarea para ejecutar la secuencia en segundo plano
          if (taskHandle == NULL) {
            xTaskCreate(iniciarTask, "IniciarTask", 2048, NULL, 1, &taskHandle);
          }
        }
        // Verificar si el string es "Parar"
        else if (received == "Parar") {
          Serial.println("Comando recibido para parar.");
          running = false; // Detener la ejecución
          if (taskHandle != NULL) {
            vTaskDelete(taskHandle); // Detener la tarea si está en ejecución
            taskHandle = NULL; // Resetear el manejador de tarea
          }
          esp_restart(); // Reiniciar el ESP32
        }
        else {
          Serial.println("El comando no es 'Iniciar' ni 'Parar'.");
        }
      }
      k++;
    } else {
      Serial.println("El vector está lleno.");
    }
  }
}

void iniciarTask(void *pvParameters) {
  digitalWrite(2, HIGH); // Encender el pin 2
  Serial.println("El pin 2 ha sido activado.");

  // Calcular velocidad
  int s = 100 * 5 / p[0];
  Serial.println(s);
```

```
102
103   // Secuencia de movimiento del servo codo
104   moverServo(servo1, p[1], p[2], s, true);
105
106   // Secuencia de movimiento del servo muñeca
107   moverServo(servo2, p[3], p[4], s, false);
108
109   digitalWrite(2, LOW); // Apagar el pin 2
110   Serial.println("Secuencia de servos completada.");
111
112   esp_restart(); // Reiniciar el ESP32 al finalizar la secuencia
113
114   vTaskDelete(NULL); // Terminar la tarea (aunque nunca se alcanzará esta línea)
115 }
116
117 // Función para mover servos
118 void moverServo(Servo& servo, int angle1, int angle2, int delayTime, bool isCodo) {
119   for (int r = 1; r <= p[5]; r++) {
120     Serial.println(r);
121
122     for (int pos = ai; pos <= ai + angle1; pos += 1) {
123       if (!running) {
124         Serial.println("Ejecución detenida.");
125         return; // Salir de la función si se ha detenido
126       }
127       servo.write(pos);
128       delay(delayTime);
129     }
130     delay(3000); // Esperar en la posición final
131
132     for (int pos = ai + angle1; pos >= ai; pos -= 1) {
133       if (!running) {
134         Serial.println("Ejecución detenida.");
135         return; // Salir de la función si se ha detenido
136       }
137       servo.write(pos);
138       delay(delayTime);
139     }
140     //delay(1000); // Esperar en la posición final
141
142     for (int pos = ai; pos >= ai - angle2; pos -= 1) {
143       if (!running) {
144         Serial.println("Ejecución detenida.");
145         return; // Salir de la función si se ha detenido
146       }
147       servo.write(pos);
148       delay(delayTime);
149     }
150     delay(3000); // Esperar en la posición final
151
152     for (int pos = ai - angle2; pos <= ai; pos += 1) {
153       if (!running) {
154         Serial.println("Ejecución detenida.");
155         return; // Salir de la función si se ha detenido
156       }
157       servo.write(pos);
158       delay(delayTime);
159     }
160     //delay(1000); // Esperar en la posición final
161   }
162 }
```