Código utilizado:

Introdução

Este código é um programa Arduino que controla seis servomotores usando comandos recebidos via comunicação serial. Cada servomotor representa uma parte de uma mão robótica (polegar, indicador, médio, anelar, mínimo e pulso), e os comandos permitem mover esses servos em diferentes incrementos.

Estrutura do Código

Bibliotecas e Variáveis

- **Servo.h**: Biblioteca usada para controlar os servomotores.
- **Variáveis Globais**:
    - `input`: Armazena o comando recebido via serial.
    - `stepSize` e `stepSize2`: Definem os tamanhos dos passos para os movimentos grandes e pequenos dos servos.
    - **Objetos Servo**: Seis objetos do tipo `Servo` para cada dedo e o pulso.

Função `setup()`

- **Inicialização dos Servos**: Cada servo é associado a um pino específico do Arduino.
- **Inicialização da Serial**: A comunicação serial é iniciada com uma taxa de transmissão de 9600 bps.

Função `loop()`

- **Leitura de Comandos**: Verifica se há algum comando disponível na serial. Se houver, chama a função `processInput()`.

Função `processInput()`

- **Leitura das Posições Atuais**: Lê as posições atuais de cada servo.
- **Processamento dos Comandos**: Ajusta as posições dos servos com base no comando recebido.
    - Cada caso no `switch` corresponde a um comando específico que altera a posição de um servo em um determinado passo.
- **Constraining**: Assegura que as posições dos servos permaneçam dentro do intervalo de 0 a 180 graus.

- **Atualização das Posições**: Chama a função `updateServoPosition()` para mover os servos para as novas posições, se necessário.

Função `updateServoPosition()`

- **Verificação e Atualização**: Verifica se a posição desejada é diferente da posição atual do servo. Se for, atualiza a posição e imprime um log na serial.

Comandos de Controle

- **Comandos para Movimentos Grandes**: (Incremento de 6 graus)
  - `c` / `v`: Move o polegar para cima / baixo.
  - `e` / `d`: Move o indicador para cima / baixo.
  - `z` / `s`: Move o médio para cima / baixo.
  - `a` / `q`: Move o anelar para cima / baixo.
  - `f` / `g`: Move o mínimo para cima / baixo.
  - `r` / `t`: Move o pulso para cima / baixo.
- **Comandos para Movimentos Pequenos**: (Incremento de 2 graus)
  - `n` / `b`: Move o polegar para cima / baixo.
  - `i` / `k`: Move o indicador para cima / baixo.
  - `o` / `l`: Move o médio para cima / baixo.
  - `p` / `m`: Move o anelar para cima / baixo.
  - `h` / `j`: Move o mínimo para cima / baixo.
  - `u` / `y`: Move o pulso para cima / baixo.

Código completo:

```
#include <Servo.h>


char input = ' ';

int stepSize = 6;

int stepSize2 = 2;


Servo thumbServo;

Servo indexServo;

Servo majeureServo;

Servo ringfingerServo;
```

```
Servo littlefingerServo;
Servo wristServo;

// Declaração da função updateServoPosition antes de seu uso
void updateServoPosition(Servo &servo, int pos, const String &name);

void setup() {
    thumbServo.attach(2);
    indexServo.attach(3);
    majeureServo.attach(4);
    ringfingerServo.attach(5);
    littlefingerServo.attach(6);
    wristServo.attach(7);
    Serial.begin(9600);
}

void loop() {
    if (Serial.available()) {
        input = (char)Serial.read();
        processInput();
    }
}

void processInput() {
    if (input == ' ') {
        return;
```

```
}

int tPos = thumbServo.read();
int iPos = indexServo.read();
int mPos = majeureServo.read();
int rPos = ringfingerServo.read();
int lPos = littlefingerServo.read();
int wPos = wristServo.read();

switch (input) {
    case 'c': tPos += stepSize; break;
    case 'v': tPos -= stepSize; break;
    case 'e': iPos += stepSize; break;
    case 'd': iPos -= stepSize; break;
    case 'z': mPos += stepSize; break;
    case 's': mPos -= stepSize; break;
    case 'a': rPos += stepSize; break;
    case 'q': rPos -= stepSize; break;
    case 'f': lPos += stepSize; break;
    case 'g': lPos -= stepSize; break;
    case 'r': wPos += stepSize; break;
    case 't': wPos -= stepSize; break;
    case 'n': tPos += stepSize2; break;
    case 'b': tPos -= stepSize2; break;
    case 'i': iPos += stepSize2; break;
    case 'k': iPos -= stepSize2; break;
```

```
        case 'o': mPos += stepSize2; break;
        case 'l': mPos -= stepSize2; break;
        case 'p': rPos += stepSize2; break;
        case 'm': rPos -= stepSize2; break;
        case 'h': lPos += stepSize2; break;
        case 'j': lPos -= stepSize2; break;
        case 'u': wPos += stepSize2; break;
        case 'y': wPos -= stepSize2; break;
    }

    tPos = constrain(tPos, 0, 180);
    iPos = constrain(iPos, 0, 180);
    mPos = constrain(mPos, 0, 180);
    rPos = constrain(rPos, 0, 180);
    lPos = constrain(lPos, 0, 180);
    wPos = constrain(wPos, 0, 180);

    updateServoPosition(thumbServo, tPos, "tPos");
    updateServoPosition(indexServo, iPos, "iPos");
    updateServoPosition(majeureServo, mPos, "mPos");
    updateServoPosition(ringfingerServo, rPos, "rPos");
    updateServoPosition(littlefingerServo, lPos, "lPos");
    updateServoPosition(wristServo, wPos, "wPos");

    input = ' ';
}
```

```
void updateServoPosition(Servo &servo, int pos, const String &name) {
    if (pos != servo.read()) {
        Serial.println("writing " + name + ": " + String(pos));
        servo.write(pos);
    }
}
```

Conclusão

Este código permite controlar os servomotores de uma mão robótica de forma precisa e eficiente, usando comandos enviados via serial. A modularização das funções e o uso de incrementos variáveis proporcionam uma flexibilidade considerável no controle dos movimentos.