```
/*
 *
================================================================================
=====
 *                        NOME DO FICHEIRO: MotorDriver_PT.ino
 *
================================================================================
=====
 *
 * DESCRIÇÃO PARA PRINCIPIANTES:
 * Este é um programa para um Arduino Mega. O Arduino é como o "cérebro"
eletrónico
 * que controla o braço robótico.
 *
 * O que este programa faz:
 * 1. Recebe ordens do "cérebro de visão" (o Raspberry Pi ou computador).
 * 2. As ordens vêm em forma de mensagem de texto (ex: "$1,0,1,1,1,1,1,90").
 * 3. O Arduino lê essa mensagem e traduz para movimentos físicos nos motores.
 *
 * TIPOS DE MOTORES USADOS:
 * - Servo Motores (controlados pela placa PCA9685): Usados para os dedos,
cotovelo e rotação da mão.
 * - Motor de Passo/Stepper (controlado por outro Arduino Uno): Usado para rodar
a base do braço.
 *
 * LINGUAGEM: C++ (Linguagem padrão do Arduino)
 *
================================================================================
=====
 */

// --- BIBLIOTECAS (Livrarias de código) ---
// Estas linhas "chamam" código extra que já foi escrito por outras pessoas
// para nos ajudar a falar com componentes específicos.
#include <Wire.h>                    // Permite a comunicação I2C (usada para
falar com a placa dos servos)
#include <Adafruit_PWMServoDriver.h>  // Biblioteca específica para controlar a
placa de servos (PCA9685)

// Criar o "objeto" que controla os servos. Imagine isto como contratar um gestor
para os motores.
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

// --- CALIBRAÇÃO (Ajustes mecânicos) ---
```

```
// Estes valores definem os limites físicos dos motores para não partirem o
braço.
// Um servo motor move-se com base num sinal de "pulso".
#define SERVOMIN  100  // Valor do pulso para o motor ir para a posição MÍNIMA
#define SERVOMAX  500  // Valor do pulso para o motor ir para a posição MÁXIMA

// --- VARIÁVEIS GLOBAIS ---
// Gavetas na memória para guardar números que vamos usar muitas vezes.
int PULSO_ABERTO;  // Guardará o valor necessário para ABRIR a mão (90 graus)
int PULSO_FECHADO; // Guardará o valor necessário para FECHAR a mão (0 graus)

// --- DEFINIÇÃO DE PINOS / PORTAS (Onde ligamos os cabos) ---
// Placa PCA9685 tem várias portas numeradas de 0 a 15.
// Aqui damos nomes a esses números para ser mais fácil de ler o código.
#define PORTA_COTOVELO 0   // O motor do cotovelo está ligado na porta 0
#define PORTA_DEDO1    13  // Dedo Polegar
#define PORTA_DEDO2    11  // Dedo Indicador
#define PORTA_DEDO3    10  // Dedo Médio
#define PORTA_DEDO4    15  // Dedo Anelar
#define PORTA_DEDO5    14  // Dedo Mínimo
#define PORTA_ROTACAO  12  // O motor que roda o pulso

// --- COMUNICAÇÃO (Conversa entre máquinas) ---
// Variáveis para receber texto do Raspberry Pi
char g_bufferPC[100];          // Uma "frase" pode ter até 100 letras
bool g_mensagemPronta = false; // "Bandeira" que sobe quando uma frase completa
chegou
int g_bufferIndex = 0;         // Contador para saber em que letra vamos

/*
 *
================================================================================
=====
 * FUNÇÃO SETUP (Configuração Inicial)
 * Esta função corre apenas UMA VEZ quando ligamos o Arduino.
 * Serve para preparar tudo antes de começar a trabalhar.
 *
================================================================================
=====
 */
void setup() {
  // 1. Iniciar Comunicações (Canais de Rádio)
  // Serial:  Ligação ao computador (cabo USB) para ver mensagens de erro no ecrã
(Debug).
  // Serial2: Ligação ao Arduino Uno (que controla a base rotativa).
```

```
  // Serial3: Ligação ao Raspberry Pi (o cérebro da visão).

  Serial.begin(115200);   // Velocidade rápida para o PC
  Serial2.begin(9600);    // Velocidade normal para o Arduino Uno
  Serial3.begin(115200);  // Velocidade rápida para o Raspberry Pi

  // 2. Iniciar o Controlador de Servos (PCA9685)
  pwm.begin();           // Acorda a placa de servos
  pwm.setPWMFreq(50);    // Define a frequência para 50Hz (padrão para servos
analógicos)
  Wire.setClock(400000); // Acelera a comunicação I2C para responder mais
depressa

  // 3. PRÉ-CÁLCULO DE POSIÇÕES
  // A função 'map' traduz graus (0-180) para valores de pulso do motor (100-
500).
  // Estamos a calcular isto agora para não ter de fazer contas a toda a hora.
  PULSO_ABERTO  = map(0, 0, 180, SERVOMIN, SERVOMAX);   // 0 graus = Aberto
  PULSO_FECHADO = map(180, 0, 180, SERVOMIN, SERVOMAX); // 180 graus = Fechado

  delay(10); // Pequena pausa para garantir que tudo estabiliza

  // 4. Posição de Início
  moverTodos(0); // Manda todos os servos para a posição 0 (repouso)

  // Mensagens para o humano ler no ecrã do PC
  Serial.println("MEGA PRONTO (MODO TURBO 115200) + ROTACAO + DEBUG ATIVADO");
  Serial.println("Mensagens exemplo para debug: ");
  Serial.println("$1,0,1,1,1,1,1,90");
  Serial.println("Aguardando comandos...");
}

/*
 *
================================================================================
=====
 * FUNÇÃO LOOP (Ciclo Infinito)
 * Esta função repete-se para sempre, milhares de vezes por segundo.
 * Aqui é onde o Arduino "vive" e trabalha.
 *
================================================================================
=====
 */
void loop() {
  // --- PARTE 1: ESCUTAR O RASPBERRY PI ---
```

```
  // Enquanto houver dados a chegar do Pi e a mensagem não estiver terminada...
  while (Serial3.available() > 0 && !g_mensagemPronta) {
    char inChar = Serial3.read(); // Lê uma letra

    if (inChar == '$') {
      // O símbolo '$' marca o INÍCIO de uma nova mensagem.
      // Se virmos isto, reiniciamos o contador para começar a escrever do zero.
      g_bufferIndex = 0;
    }
    else if (inChar == '\n') {
      // O símbolo '\n' (mudança de linha) marca o FIM da mensagem.
      g_bufferPC[g_bufferIndex] = '\0'; // Fecha a "string" (texto) corretamente
      g_mensagemPronta = true;          // Levanta a bandeira: "Temos uma ordem!"
    }
    else if (g_bufferIndex < 99) {
      // Se não for início nem fim, é conteúdo (ex: '1', ',', '9').
      // Guardamos no buffer e avançamos o contador.
      g_bufferPC[g_bufferIndex] = inChar;
      g_bufferIndex++;
    }
  }

  // --- PARTE 2: EXECUTAR A ORDEM ---
  // Se a bandeira estiver levantada (mensagem completa recebida)...
  if (g_mensagemPronta) {
    processarMensagemRapida(); // Chama a função que distribui as tarefas
    g_mensagemPronta = false;  // Baixa a bandeira e prepara para a próxima
    g_bufferIndex = 0;
  }
}

/*
 *
================================================================================
=====
 * FUNÇÃO: processarMensagemRapida
 * Objetivo: Pegar no texto recebido e comandar cada motor individualmente.
 * Formato da mensagem esperada: "$orient,flex,d1,d2,d3,d4,d5,rotacao"
 *
 * Exemplo: "1,0,1,1,1,1,1,90"
 * Significa:
 *   - Orientação (Base): 1
 *   - Flexão (Cotovelo): 0
 *   - Dedos (1-5): Todos 1 (fechados/abertos dependendo da lógica)
 *   - Rotação: 90 graus
```

```
 *
================================================================================
=====
 */
void processarMensagemRapida() {
  // DEBUG: Mostra no ecrã do PC o que o Arduino acabou de ouvir
  Serial.print("RX (Recebido): ");
  Serial.println(g_bufferPC);

  // A função 'strtok' serve para "partir" o texto sempre que encontra uma
vírgula.

  // --- PASSO 1: BASE (Arduino Uno) ---
  // Pega na primeira parte do texto (antes da primeira vírgula)
  char* token = strtok(g_bufferPC, ",");
  if (token == NULL) return; // Se não houver nada, cancela (segurança)

  // Envia essa ordem para o outro Arduino (Uno) que controla a base
  Serial2.write(token[0]);
  Serial.print("BASE -> Uno: ");
  Serial.println(token[0]);

  // --- PASSO 2: SERVOS (Mão e Cotovelo) ---
  // Para cada componente, lemos o próximo pedaço do texto e movemos o motor.

  // Cotovelo (Digital: 0 ou 1)
  moverDigital(PORTA_COTOVELO);

  // Dedos (Digital: 0 ou 1)
  moverDigital(PORTA_DEDO1); // Polegar
  moverDigital(PORTA_DEDO2); // Indicador
  moverDigital(PORTA_DEDO3); // Médio
  moverDigital(PORTA_DEDO4); // Anelar
  moverDigital(PORTA_DEDO5); // Mínimo

  // --- PASSO 3: ROTAÇÃO DO PULSO (Analógico: 0 a 180 graus) ---
  moverAnalogico(PORTA_ROTACAO);
}

/*
 *
================================================================================
=====
 * FUNÇÃO: moverDigital
 * Objetivo: Controlar motores que só têm dois estados (Aberto/Fechado).
```

```
 * Argumentos: 'porta' (qual o motor que vamos mexer).
 *
================================================================================
=====
 */
inline void moverDigital(int porta) {
  // 'strtok(NULL, ",")' pede o PRÓXIMO pedaço de texto após a última vírgula
encontrada
  char* token = strtok(NULL, ",");

  if (token != NULL) {
    // Verifica se a ordem é "1" ou "0"
    // Sintaxe: (condição) ? valor_se_verdadeiro : valor_se_falso
    int pulso = (token[0] == '1') ? PULSO_FECHADO : PULSO_ABERTO;

    // Manda a ordem para a placa de servos
    pwm.setPWM(porta, 0, pulso);

    // Mensagem de debug para o técnico ver no PC
    Serial.print("D["); Serial.print(porta); Serial.print("]: ");
    Serial.print(token[0] == '0' ? "FECHADO" : "ABERTO");
    Serial.print(" ("); Serial.print(pulso); Serial.println(")");
  }
}

/*
 *
================================================================================
=====
 * FUNÇÃO: moverAnalogico
 * Objetivo: Controlar motores que podem estar em qualquer ângulo (0-180).
 * USADO PARA: Rotação do pulso.
 *
================================================================================
=====
 */
inline void moverAnalogico(int porta) {
  char* token = strtok(NULL, ",");

  if (token != NULL) {
    // 'atoi' converte texto ("123") para número inteiro (123)
    int angulo = atoi(token);

    // SEGURANÇA: Impedir que o motor tente ir além dos limites físicos
    if (angulo < 0) angulo = 0;
```

```
    if (angulo > 180) angulo = 180;

    // Traduz o ângulo (0-180) para o "idioma" do motor (pulsos)
    int pulso = map(angulo, 0, 180, SERVOMIN, SERVOMAX);

    // Executa o movimento
    pwm.setPWM(porta, 0, pulso);

    Serial.print("ROT["); Serial.print(porta); Serial.print("]: ");
    Serial.print(angulo);
    Serial.print(" deg ("); Serial.print(pulso); Serial.println(")");
  }
}

/*
 *
================================================================================
=====
 * FUNÇÃO: moverTodos
 * Objetivo: Mover todos os motores para um ângulo específico de uma vez.
 * Útil para inicialização.
 *
================================================================================
=====
 */
void moverTodos(int ang) {
  int pulso = map(ang, 0, 180, SERVOMIN, SERVOMAX);
  // Loop 'for': Repete para cada motor de 0 a 6
  for (int i = 0; i <= 6; i++) {
    pwm.setPWM(i, 0, pulso);
  }
}
```