|  |  |
| --- | --- |
| #include <SpeedyStepper.h> | Faz a inclusão da biblioteca Speedy Stepper, responsável pelo controle dos motores de passo. |
| const int MOTOR\_X\_STEP\_PIN = 2;  const int MOTOR\_Y\_STEP\_PIN = 3;  const int MOTOR\_Z\_STEP\_PIN = 4;  const int MOTOR\_X\_DIR\_PIN = 5;  const int MOTOR\_Y\_DIR\_PIN = 6;  const int MOTOR\_Z\_DIR\_PIN = 7;  const int STEPPERS\_ENABLE\_PIN = 8;  const int LIMIT\_SWITCH\_X\_PIN = 9;  const int LIMIT\_SWITCH\_Y\_PIN = 10;  const int LIMIT\_SWITCH\_Z\_PIN = 11; | Faz a atribuição dos pinos digitais do Arduino, definindo sua responsabilidade no código.  Neste caso define-se os responsáveis pelo controle dos passos dos motores, assim como da direção dos mesmos.  Também é definido o pino de habilitação de movimento dos motores, e os sensores de fim de curso necessários para zerar a posição dos motores. |
| SpeedyStepper stepperX;  SpeedyStepper stepperY;  SpeedyStepper stepperZ; | Faz-se a instanciação dos motores (nomeadamente motores X, Y e Z), a partir de modelo definido pela biblioteca Speedy Stepper. |
| long int diferenca\_tempo = 0;  unsigned long tempo\_1 = 0;  unsigned long tempo\_2 = 0;  int num\_ext;  int num\_des;  int conf = 1;  long int trajeto\_volta;  long int movimento\_x = 533\*15;  long int movimento\_y = 533\*20;  long int movimento\_z = 533\*42;  long int movimento\_y\_curto = 533\*12;  long int movimento\_z\_curto = 533\*15;  long int movimento\_y\_restante = movimento\_y - movimento\_y\_curto;  long int movimento\_z\_restante = movimento\_z - movimento\_z\_curto;  int trajeto\_x = 533;  int trajeto\_y = 533;  int trajeto\_z = 533; | Atribuição de variáveis necessárias para o funcionamento do programa. Estas variáveis servem para questões de controle de tempo total de execução, confirmação de dados com o usuário, e constantes relacionadas ao número de passos que os motores precisam realizar para movimentação em mm. |
| void setup() {  Serial.begin(9600);  pinMode(STEPPERS\_ENABLE\_PIN, OUTPUT);  pinMode(LIMIT\_SWITCH\_X\_PIN, INPUT\_PULLUP);  pinMode(LIMIT\_SWITCH\_Y\_PIN, INPUT\_PULLUP);  pinMode(LIMIT\_SWITCH\_Z\_PIN, INPUT\_PULLUP);  stepperX.connectToPins(MOTOR\_X\_STEP\_PIN, MOTOR\_X\_DIR\_PIN);  stepperY.connectToPins(MOTOR\_Y\_STEP\_PIN, MOTOR\_Y\_DIR\_PIN);  stepperZ.connectToPins(MOTOR\_Z\_STEP\_PIN, MOTOR\_Z\_DIR\_PIN);  digitalWrite(STEPPERS\_ENABLE\_PIN, HIGH);  stepperX.setSpeedInStepsPerSecond(1350);  stepperX.setAccelerationInStepsPerSecondPerSecond(400);  stepperY.setSpeedInStepsPerSecond(500);  stepperY.setAccelerationInStepsPerSecondPerSecond(100);  stepperZ.setSpeedInStepsPerSecond(1350);  stepperZ.setAccelerationInStepsPerSecondPerSecond(400);  } | Estas instruções são executadas uma única vez, no início do programa. Aqui vão configurações dos pinos do Arduino e de seu funcionamento.  Inicialmente se inicia a comunicação serial, a 9600 bps. Isso será responsável pela comunicação da placa com o usuário, através do monitor serial, onde serão inseridas informações de número de ciclos de extração e de dessorção.  Faz-se a declaração dos pinos referentes aos fins de curso, do tipo INPUT\_PULLUP, para evitar interferências de sinal através dos fios, o que poderia acarretar mal funcionamento do programa (verificação incorreta de posição). Define-se também os pinos para cada um dos motores instanciados anteriormente, e faz-se a definição de condições de movimento para cada um deles: velocidade e aceleração. |
| void mover\_x(long int x) {  stepperX.moveRelativeInSteps(x);  delay(100);  }  void mover\_y(long int y) {  stepperY.moveRelativeInSteps(y);  delay(100);  }  void mover\_z(long int z) {  stepperZ.moveRelativeInSteps(z);  delay(100);  } | Criação de funções responsáveis pelo movimento dos motores. Cada função recebe como parâmetro um número de passos, que é passado para cada motor instanciado, que então executa o comando, movimentando seu respectivo eixo. |
| void extrair(int num\_ext) {  Serial.println("Executando extração... \n");  for(int n = 0; n < num\_ext; n++) {  mover\_x(-movimento\_x);  mover\_z(-movimento\_z);  mover\_y(movimento\_y\_curto);  delay(2500);  mover\_z(movimento\_z\_curto);  mover\_y(movimento\_y\_restante);  delay(1000);  mover\_y(-movimento\_y);  mover\_z(movimento\_z\_restante);  }  } | Esta seção de código cria uma função responsável por executar os ciclos de extração, recebendo como parâmetro o número de ciclos informado pelo usuário.  Ela então executa uma estrutura de repetição, que faz o movimento de extração pelo número de vezes necessário. |
| void dessorver(int num\_des) {  Serial.println("Executando dessorção... \n");  mover\_x(-movimento\_x);  mover\_z(-movimento\_z);  for (int n = 0; n < num\_des; n++) {  mover\_y(movimento\_y\_curto);  delay(2500);  mover\_z(movimento\_z\_curto);  mover\_y(movimento\_y\_restante);  delay(1000);  mover\_y(-movimento\_y);  if (n < (num\_des-1)) {  mover\_z(-movimento\_z\_curto);  } else {  mover\_z(movimento\_z\_restante);  }  }  } | Já esta seção é semelhante a anterior, mas desta vez responsável pela execução dos ciclos de dessorção. De mesma forma, a partir de número informado pelo usuário, executa os ciclos de dessorção. |
| void inicia\_tempo() {  tempo\_1 = millis()/1000;  digitalWrite(STEPPERS\_ENABLE\_PIN, LOW);  delay(50);  } | A função inicia\_tempo é responsável por iniciar a contagem de tempo, para que seja possível ao final informar ao usuário o tempo total |
| void termina\_tempo() {  digitalWrite(STEPPERS\_ENABLE\_PIN, HIGH);  delay(50);  tempo\_2 = millis()/1000;  diferenca\_tempo = tempo\_2 - tempo\_1;  int minutos = (diferenca\_tempo/60)%60;  int segundos = diferenca\_tempo%60 + 1;  Serial.print("O tempo total aproximado foi: ");  Serial.print(minutos); Serial.print(" min e ");  Serial.print(segundos); Serial.println(" segundos.");  } | Funcionando de forma complementar a função inicia\_tempo, a função termina\_tempo é responsável por terminar a contagem de tempo, calcular o tempo total gasto, e informar o usuário esta informação. |
| void executa (int num\_ext, int num\_des) {  extrair(num\_ext);  dessorver(num\_des);  trajeto\_volta = (num\_ext+1)\*movimento\_x;  Serial.println("Voltando à posição de origem \n");  mover\_x(trajeto\_volta);  } | A função executa é responsável por realizar o processo, passando informações de extração e de dessorção para suas respectivas funções, a por fazer o movimento final, retornando o porta amostra para a posição de origem |
| void pergunta1() {  Serial.println("\nPor favor, indique o número de ciclos de extração desejado:");  while (Serial.available() == 0) {}  num\_ext = Serial.parseInt();  Serial.print("Você selecionou: ");  Serial.println(num\_ext);  delay(500);  } | Esta função é responsável pela interação com o usuário, perguntando qual o número de ciclos de extração desejado. |
| void pergunta2() {  Serial.println("\nAgora, indique o número de ciclos de dessorção desejado:");  while (Serial.available() == 0) {}  num\_des = Serial.parseInt();  Serial.print("Você selecionou: ");  Serial.println(num\_des);  delay(500);  } | De forma similar a anterior, esta função interage com o usuário fazendo a pergunta de qual o número de ciclos de dessorção desejado. |
| void confirmacao() {  if (num\_ext>9 || num\_ext<1) {  Serial.println("Você escolheu um número não permitido de ciclos de extração. Escolha um valor de 1 a 9. \n");  principal();  }  if (num\_des>9 || num\_des<1) {  Serial.println("Você escolheu um número não permitido de ciclos de dessorção. Escolha um valor de 1 a 9. \n");  principal();  } else {  Serial.print("Você selecionou ");  Serial.print(num\_ext); Serial.print(" ciclo(s) de extração(ões) e ");  Serial.print(num\_des);  Serial.println(" ciclo(s) de dessorção(ões). \n\nPara corrigir digite 0. Para iniciar digite 1 \n");  while (Serial.available() == 0) {}  conf = Serial.parseInt();  if (conf>1 || conf<0) {  Serial.println("Você escolheu um número não permitido. Favor inserir 1 para confirmar e 0 para cancelar: \n");  while (Serial.available() == 0) {}  conf = Serial.parseInt();  confirmacao();  }  }  } | Função de controle de dados informados pelo usuário. Dadas limitações físicas do protótipo, verifica se os números de ciclo de extração e de dessorção estão entre 1 e 9.  Também permite que o usuário revise os dados informados antes de executar o processo. |
| void zerar\_posicao(){  const long directionTowardHome = 1;  const float speedInStepsPerSecond = 100;  const long maxDistanceToMoveInSteps = 9999999;  stepperZ.moveToHomeInSteps(directionTowardHome, 15\*speedInStepsPerSecond, maxDistanceToMoveInSteps, LIMIT\_SWITCH\_Z\_PIN);  Serial.println("Posição Z verificada.\n");  delay(100);  stepperY.moveToHomeInSteps(-directionTowardHome, 8\*speedInStepsPerSecond, maxDistanceToMoveInSteps, LIMIT\_SWITCH\_Y\_PIN);  Serial.println("Posição Y verificada.\n");  delay(100);  stepperX.moveToHomeInSteps(directionTowardHome, 20\*speedInStepsPerSecond, maxDistanceToMoveInSteps, LIMIT\_SWITCH\_X\_PIN);  Serial.println("Posição X verificada.\n");  delay(100);  mover\_x(-500);  mover\_y(3\*533);  mover\_z(-13\*533);  } | Esta função é responsável por colocar o protótipo na posição correta antes de iniciar o processo de extração. Isso é importante pois entre usos, os eixos podem ser movimentados manualmente. Assim, independentemente da posição em que o protótipo é deixado, o programa é responsável por colocar ele na posição correta através dos limitadores de fim de curso instalados. |
| void principal() {  Serial.println("Olá, bem-vindo!");  delay(1000);  if (conf == 1) {  pergunta1();  delay(250);  pergunta2();  delay(250);  confirmacao();  if (conf == 0) {  Serial.println("\nOK! Vamos começar novamente. \n");  delay(2500);  conf = 1;  principal();  }  if (conf == 1) {  Serial.println("Iniciando o procedimento... \n");  delay(500);  Serial.println("Fazendo verificação do aparelho...\n");  inicia\_tempo();  zerar\_posicao();  executa(num\_ext, num\_des);  termina\_tempo();  Serial.println("\nObrigado por utilizar este programa!");  Serial.end();  }  }  } | Principal função do programa, centraliza todas as ações, desde iniciar contato com o usuário para obter informações, até chamar os contadores de tempo, zerar posição dos eixos, executar todo o processo e se despedir do usuário. |
| void loop(){  principal();  } | Inicia automaticamente a função principal. |