Resposta da Questão 1: (SEM GIT)

Alexandre Nuernberg

Uma empresa farmacêutica de manipulação de medicamentos realiza a mistura de duas substâncias para gerar um determinado remédio. Essa mistura precisa ser realizada em condições específicas de temperatura e velocidade de agitação, para gerar a consistência e homogeneidade necessárias.

Atualmente, esse processo é feito de forma manual, onde um funcionário é responsável por realizar a agitação das duas substâncias em uma caldeira, enquanto controla a intensidade de uma chama para manter a mistura aquecida.

Como o processo é demorado e custoso, manter uma pessoa por muito tempo exclusivamente para essa atividade não se torna mais viável. Pensando nisso, a empresa farmacêutica fechou um contrato para um projeto de P&D com o objetivo de automatizar o processo. Abaixo, requisitos para sua implementação:

- Duração do processo: 120 min:
  - Durante os primeiros 30 min:
    - Velocidade de rotação da mistura: 20 rpm;
    - Temperatura da mistura: 50 ºC.
  - De 30 a 60 min:
    - Velocidade de rotação da mistura: 30 rpm;
    - Temperatura da mistura: 65 ºC.
  - De 60 a 120 min:
    - Velocidade de rotação da mistura: 40 rpm;
    - Temperatura da mistura: 80 ºC.

A partir dos requisitos de projeto para o misturador, considere:

- Considere os entregáveis:
  a. Levantamento de arquitetura de software embarcado;
  b. Especificação de microcontrolador/processador;
  c. Especificação de periféricos necessários:
     i. Entradas para leitura;
     ii. Saídas para atuação.
  d. Geração de um pseudocódigo e fluxograma para a solução, considerando os requisitos do processo indicados anteriormente.

- Gere um PDF com o texto explicativo da solução proposta, contendo os entregáveis listados.

SER O MAIS ESPECÍFICO POSSÍVEL PARA ESSA SOLUÇÃO.
PODE ANEXAR O PDF NO REPOSITÓRIO.

1. Introdução:

A produção de medicamentos no Brasil é um processo altamente regulamentado para garantir a segurança e a eficácia dos produtos. As indústrias farmacêuticas devem se conformar a uma série de normativas emitidas principalmente pela Anvisa (Agência Nacional de Vigilância Sanitária), que é o órgão regulador do setor.

Entre essas normativas, no que tange a produção de medicamentos, podemos citar:

Boas Práticas de Fabricação (BPF): A RDC nº 658/2022 estabelece as diretrizes gerais de BPF para medicamentos, incluindo requisitos para instalações, equipamentos, pessoal, controle de qualidade, documentação e validação de processos. As BPF são essenciais para garantir a qualidade dos medicamentos e prevenir riscos à saúde.

A RDC 658/2022 não menciona diretamente a compatibilidade de suas normas técnicas com órgãos reguladores do setor sanitário de outros países.

Para a elaboração desse projeto, levando-se em conta que é uma operação que exige normatização sanitária, foi considerado o uso de um sensor de temperatura, que ficará em contato com o produto, esse sensor atende tanto as normas norte americanas quanto europeias conforme detalhado em sua especificação técnica.

Para a resolução da questão foram elencados diversas etapas que serão descritas na sequência e alguns requisitos técnicos que não foram definidos no enunciado foram considerados de forma a se poder delinear uma solução técnica.

2. Metodologia:

2.1 Definição dos requisitos técnicos:

Para a elaboração do seguinte projeto alguns requisitos técnicos, que não foram previamente citados no enunciado serão assumidos de forma a permitir o desenvolvimento de uma solução.

Para questões de dimensionamento dos motores controladores e do controlador de chamas, está se assumindo que o tanque de processamento (caldeira) é de 50 l. Também foi assumido que não há limitação de espaço físico para a instalação dos instrumentos e equipamentos. Não há limitação de custos definida.

2.2 (a) Levantamento de arquitetura de software embarcado:

Para o controle, optou-se pela utilização de um microcontrolador industrial de um fabricante consolidado no mercado ST Microelectronics e que tem a quantidade de portas de entrada/saída e periféricos como conversores analógico digital e digital analógico bem como portas com saída PWM além de interface SPI, compatíveis com a necessidade do projeto.

A programação será desenvolvida em linguagem “C”, que é compatível com o microcontrolador em questão, além de ser robusta e consolidada no mercado. O microcontrolador escolhido pertence à família ARM Cortex-M4 de 32 bits que possui desempenho adequado para a realização das tarefas propostas que são elencadas abaixo:

1. Controle do motor:
    a. Leitura da velocidade RPM pelo encoder
    b. Ajuste da velocidade de rotação do motor atuando no driver do motor
2. Controle da Temperatura:
    a. Medição da temperatura do sensor

   b. Ajuste do controlador de chamas
3. Alertas:
   a. ativação de alarme sonoro se estados críticos atingidos ou parâmetros diferentes dos estipulados
4. Emergência
   a. botão de emergência para o desligamento do motor e controlador de chamas
5. Interface Homem-Máquina (IHM)
   a. Display para visualização do dashboard com as informações da operação (Temperatura, velocidade do motor, nível da chama, alarmes, tempo do processo).

A figura 1 mostra o diagrama de blocos simplificado da solução proposta, onde podem ser vistos os principais elementos que compões a solução

*Para a solução haverá a necessidade do uso de vários níveis de tensão (+5Vcc, +12Vcc e 48Vcc), e as fontes ou conversores não foram detalhados no diagrama.

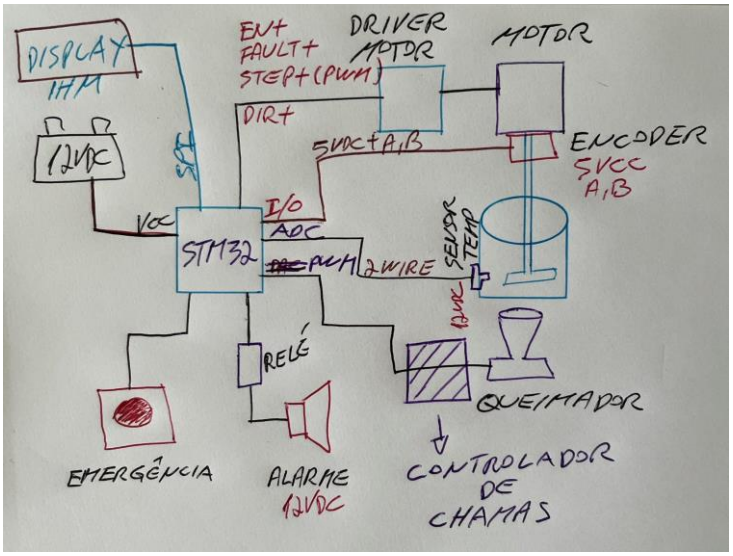

Figura 1 - Diagrama de blocos com os principais componentes da solução

2.3 (b) Especificação de microcontrolador/processador:
O microcontrolador é o cérebro da operação, nele ficará armazenado o código que irá efetuar o controle das atividades, desde a verificação da temperatura no sensor de temperatura, a correção da mesma, através do controle do controlador da chama. Através do módulo de encoder a rotação do motor poderá ser medida e posteriormente corrigida através do driver que controla o motor. Além disso um módulo para segurança monitora o status do botão de emergência e status do driver do motor, para situações críticas em que o montor ou o controlador de chamas devam ser desligados e um modo de alerta poderá alarmar caso ocorra disparidades nos valores de temperatura, rotação ou ocorra falha no controlador do motor. E finalmente através da interface IHM, atualizar as informações do dashboard com os dados da operação. Ele também terá definido os módulos de controle das etapas onde garantiram que para cada período especificado a temperatura e a rotação devem estar controladas dentro dos parâmetros especificados.

Para o desenvolvimento do controlador foi escolhido um microcontrolador de uso industrial STM32 modelo STM32L476RG que possui as seguintes características:

- Processador: ARM Cortex-M4 de 32 bits com unidade de ponto flutuante (FPU) e frequência de clock de até 80 MHz, oferecendo alto desempenho para aplicações complexas.
- Memória: 1 MB de Flash e 128 KB de RAM

- Periféricos:
    - Diversos periféricos, incluindo timers, ADCs, DACs, USART, SPI, I2C, PWM.
- Recursos avançados:
    - DMA (Direct Memory Access) para transferências de dados eficientes.
    - Dois barramentos SPI

Para facilitar o desenvolvimento será utilizado uma placa pronta com o microcontrolador em questão 'Nucleo L476RG" [1]. Essa placa possui 2 DACs e 17 saídas PWM, DAC1: Possui 2 canais (canal 1 e canal 2), com resolução de 12 bits. DAC2: Possui 1 canal (canal 1), com resolução de 12 bits.

2.4 (c) Especificação de periféricos necessários:
Segue um descritivo dos principais periféricos que serão necessários para a implementação da solução e na sequência, serão detalhadas as entradas para a leitura e as saídas para atuação.

Misturador: é composto por um motor de passo, um driver de controle do motor e um Encoder para se medir a rotação do mesmo.

- **Motor de passo:** Motor de Passo HT34-487-R10P | Torque 130kg com Redutor Planetário 10:1 – Kalatec [2]
  Foi escolhido um motor de passo, que permite um controle bastante preciso da posição e da velocidade de rotação, incluindo a definição de rotações em RPM, além disso o motor escolhido tem torque adequado para a aplicação e uma caixa de redução uma vez que as velocidades em questão são baixas (até 40rpm).

  Características:
    - Motor de Passo Nema: 34
    - Flange: 86mm
    - Passo em ângulo: 1,80
    - Resolução: 200 PPR (Pulsos Por Rotação)
    - Torque Estático Bipolar: 13Nm (130 Kg.cm) sem a redução
    - Quantidade de Fios: 08
    - Ligações possíveis: Unipolar e Bipolar Serie e Bipolar Paralelo
    - Corrente Unipolar: 6,0A/fase
    - Diâmetro do eixo saída do redutor: 20mm com chaveta

- **Drive de Motor de Passo:** 8A | STR8 | KALATEC [3]
  O drive escolhido é do mesmo fabricante e compatível com o motor selecionado.
  O controle é feito pelo microcontrolador conectando-se uma porta PWM do STM32 ao pino STEP+ do STR8. Além disso o pino EN+ será utilizado para habilitar o driver do motor pelo microcontrolador e a rotação do motor, apesar de não ser pré-requisito, poderá ser controlada pelo pino DIR+.
  Características:
    - Corrente de 2,35 a 8,0A
    - Alimentação de 24 a 75VDC
    - Resolução: 200 / 400 / 2000 / 5000 /12800 / 20.000 PPR
    - Sinal Pulso: 5 ~ 24V (NPN / PNP)
    - Conexões EN+, Fault+, STEP+ (PWM), Dir+

- **Encoder:** Baumer ITD 40 A 4 Y79 [4]
  A medição da velocidade do motor é um ponto crítico uma vez que a mistura precisa ser realizada em condições específicas de temperatura e rotação. Para garantir que o motor esteja efetuando a mistura na velocidade correta um encoder é acoplado ao mesmo de forma a poder medir a velocidade de rotação do eixo.
  Características:
    - Voltage supply 5 VDC ±5 %
    - Pulses per revolution 200 … 2048 (200ppr)
    - Output signals A, B, N + inverted (A e B)

- Output stages TTL linedriver (short-circuit proof)

Medição da temperatura: para a medição da temperatura foi escolhido um sensor adequado ao uso a indústria farmacêutica e que o atende as normas de higiene.

- **Sensor de temperatura:** Baumer TER-8 de -40°C a 115°C. [5]
  Esse sensor é adequado ao uso de aplicações em indústrias farmacêuticas e alimentícias e tem todo o corpo coberto em aço inox especial isso se faz necessário pois ele estará em contato direto com o produto.
  O TER-8 utiliza uma saída de corrente padrão de 4-20 mA (em configuração de 2 fios). Para uma conexão analógica direta, recomenda-se utilizar essa versão de saída de 4-20 mA, que é mais comum para comunicação com microcontroladores.
  Para que o STM32 leia o sinal de 4-20 mA, será necessário utilizar um resistor de shunt para converter essa corrente em uma tensão, para operar diretamente na faixa de entrada do ADC do STM32 (0-3,3V), pode-se usar um resistor de 165 Ω, que gerará uma faixa de aproximadamente 0,66 a 3,3V.

  Características:
  - Voltage supply range 8 ... 35 V DC
  - Resistance Pt100 (4-wire)
  - Sampling interval 0.5 s
  - Max. measuring error ± 0.25 °C
  - Atende as normas de higiene
    - EN 61326-1
    - EN 61000-6-2
    - EN 61000-6-3
    - 1935/2004/EG
    - 10/2011/EU
    - 2023/2006/EG
    - FDA (21 CFR 177.2416)
    - 3-A (74-07)

Sistema de alarme: Será utilizada uma sirene industrial como alarme sonoro caso os parâmetros de alguma etapa sejam extrapolados ou para alarmar outros tipos de falhas como do controlador do motor de passo ou o acionamento do botão de emergência.

- **Sirene Eletrônica Multisom** IP54 - Baixo Consumo - **TUCSG200 [6]**

  Características:
  - Sirene com possibilidade de 32 sons diferentes
  - Pode ser utilizado em ambiente externo
  - Grau de proteção: IP65
  - Pressão sonora de até 114 dB
  - Alimentação: 12 / 24 Vdc - 90 ~ 230 Vac

- **Controlador da chama:** Kromschröder BCU 460 [7]
  Este controlador oferece diversas opções para controlar a intensidade da chama e, consequentemente, a temperatura da caldeira. É um controlador de chama com foco em segurança e confiabilidade adequado a um ambiente de produção de medicamentos.
  Recursos:
  Monitoramento de chama por ionização ou UV.
  Sequenciamento de ignição configurável.
  Funções de segurança avançadas.
  Diagnóstico de falhas detalhado.
  Comunicação: Possui entradas e saídas digitais
  Para controlar a chama através do STM32, pode-se utilizar uma saída PWM do STM32 para controlar a entrada 0-10V do BCU 460 (pino 3 do conector X1).
  É necessário fazer uma pré configuração no controlador para interpretar o sinal PWM como um valor de referência para a intensidade da chama.

- **Dashboard:** será implementado através uso de um display externo conectado ao microcontrolador, permitindo a visualização as variáveis como temperatura velocidade e tempo de execução da mistura e alarmes.
    - Display Industrial 7 polegadas com SPI [8]
    - Modelo: Riverdi RiTFT-70-WCAG01-SPI

Características:

- Tela TFT LCD de 7 polegadas com resolução de 800x480 pixels.
- Interface SPI.
- Controlador ILI9341.
- Amplo ângulo de visão (80° em todas as direções).

I. **Entradas para leitura:**

As seguintes entradas de dados serão usadas.

a. **Leitura do sensor de temperatura:** Permite a verificação da temperatura da Caldeira para efetuar os ajustes necessários de acordo com as etapas do processo. As etapas estão descritas no Quadro 1. Esse processo é feito através da conexão do sensor de temperatura a interface do conversor analógico digital do STM32. Para este processo também é necessário um resistor shunt (derivação) uma vez que o conversor analógico digital mede a variação da tensão e o sensor de temperatura fornece variação da corrente e uma conversão de corrente para tensão deve ser feita.

b. **Leitura da rotação do encoder:** o encoder gera pulsos digitais nos pinos A e B a cada incremento de posição, o STM32 precisa ler esses pulsos para determinar a velocidade do motor.

c. **Leitura da falha do driver do motor de passo (FAULT+):** o motor de passo é um componente crítico do processo ele possui um pino de saída que é ativo alto caso ocorra falha do driver esse pino será monitorado numa entrada do STM32 para acionar o alarme caso a falha ocorra.

d. **Leitura do botão de emergência:** o botão estará conectado a um pino de entrada do microcontrolador e se ele for acionado tanto o motor quanto o controlador de chamas serão desligados e um alarme sonoro será acionado bem como um alerta visual no display IHM.

II. **Saídas para atuação:** os seguintes atuadores serão acionados pelo microcontrolador.

a. **Saída do alarme (atuação do relé):** um pino de saída é utilizado para acionar o relé esse é acionamento pode ser feito através do uso de um transistor controlando diretamente o relé. Outra opção é não utilizar o relé e utilizar um transistor de potência que seja compatível com a potência da sirene.

b. **Saída dos dados no display:** as seguintes variáveis estarão disponíveis visualmente no display HID (rotação do motor, temperatura da caldeira, status de alerta do driver do motor, status do botão de emergência). Para a comunicação com o display será utilizado o Barramento SPI do microcontrolador.

c. **Driver do motor (EN+ e STEP+):** o driver do motor de passo possui 2 pinos de entrada que serão controlados pelo microcontrolador um deles é o pino de enable (EN+) que quando ativo liga o driver. O outro é o pino de STEP+, que receberá os pulsos PWM proporcionais a velocidade de rotação desejada.

d. **Saída do controlador de chamas:** o controle da abertura ou fechamento da chama será feito através de outro pino PWM do STM32.

2.5 (d) Geração de um pseudocódigo e fluxograma para a solução, considerando os requisitos do processo indicados anteriormente.

O *Quadro 1* descreve as etapas com seus respectivos tempos, temperatura de aquecimento da caldeira e velocidade de rotação do misturados.

Quadro 1: Etapas para a manufatura.

Etapas da manufatura			
Etapa	1	2	3
Temperatura (°C)	50	65	80
Rotação (rpm)	20	30	40
Tempo (min)	30	30	60

Pseudocódigo

Esse pseudocódigo possui os blocos principais, mas não entra em detalhe nas minúcias do processo de configuração dos periféricos no microcontrolador STM32 (rotinas de interrupção, barramento SPI, comunicação com o display, configurações de PWM e leituras do conversor analógico digital, nem dos cálculos para converter os valores do encoder para RPM e do ADC para temperatura, pois para isso é necessária uma maior demanda de tempo e leitura dos manuais tanto do sensor e de temperatura, quanto do driver do motor de passo e controlador de chama e display.
Nessa fase também não houve a preocupação de definição dos tipos das variáveis (int, float, boolean, string, etc).

```
/*Rotina de Inicialização*/

configura_pinos();  /*configurar pinos do STM32 (entradas, saídas,
                                  interrupções, modo PWM, interface SPI)*/
configura_ADC();     //configurar ADC para leitura do sensor de temperatura;
configura_PWM1();    //configurar timer para PWM1 do motor de passo;
configura_PWM2();    //configurar timer para PWM2 do controlador de chamas;
inicia_SPI();        //configurar comunicação SPI com o display;

/*declaração de variáveis e constantes*/
// inicializa_variáveis
temperatura;
velocidade;
setpoint;
tempos e flags das etapas;

/*Etapa 0: Fase inicial onde o motor e o queimador são ligados e os valores
inicias de trabalho (20rpm e 50°C) começam a ser incrementados e deve-se aguardars
para iniciar o processo na etapa 1 */
flag_etapa=0; //informa que o processo está na etapa inicial de pre aquecimento e
rotação
inicia_timer=0; //flag da contagem do timer
etapa(0); //inicializa a etapa inicial
mostra_display(); // mostra os dados iniciais no display

/*Loop principal*/
while (verdadeiro) { //rotina de loop infinito
      /*Rotinas da Etapa 1*/
      etapa(1); //inicializa a primeira etapa
              while (flag_etapa == 0 || flag_etapa == 1) { //só roda essa etapa para
os flags iniciais (aquecimento) e o próprio flag da etapa
              etapa(1); //inicializa a primeira etapa
              flag_etapa = 1; //informa que está na primeira etapa
              checa_timer(1); //checa o timer para a etapa 1 (30 min)
              mostra_display(); mostra os dados no display
      }
```

```
      /*Rotinas da Etapa 2*/
      etapa(2); //inicializa a segunda etapa
      while (flag_etapa == 2) //só roda essa etapa para o flag_etapa = 2
             etapa(2); //inicializa a primeira etapa
             flag_etapa = 2; //informa que está na primeira etapa
             checa_timer(2); //checa o timer para a etapa 2 (30 min)
             mostra_display(); mostra os dados no display
      }
      /*Rotinas da Etapa 3*/
      etapa(3); //inicializa a terceira etapa
      while (flag_etapa == 3)  //só roda essa etapa para o flag_etapa = 3
             etapa(3); //inicializa a primeira etapa
             flag_etapa = 3; //informa que está na primeira etapa
             checa_timer(3); //checa o timer para a etapa 3 (60 min)
             mostra_display(); mostra os dados no display
      }

      /*Rotinas da Etapa Deliga*/
      etapa(desliga); //Desliga o sistema
      while (flag_etapa == desliga)  //só roda essa etapa para o flag_etapa = 2
             flag_etapa = desliga; //informa que está na primeira etapa
             checa_timer(desliga); //checa o timer para a etapa desliga
             mostra_display(); mostra os dados no display
      }

      // Aguardar um tempo para refazer a análise
    delay(100ms);

}

/*Definição de Funções*/
//Essa função lida com a manipulação dos parâmetros específicos por etapa, detalhados
no "Quadro 1" do documento
etapa(X){
      se (X == 0) { //Etapa 0 (pré aquecimento e rotação)
             inicia_aquecimento(50);       //inicializar o aquecimento da caldeira até
50°C
             inicia_rotação(20);           //inicializar a rotação do motor até 20rpm
      }
      se (X == 1) { //inicia a etapa 1 T=50 e RPM=20
             checa_temperatura(50); //verifica se a temperatura atingiu 50°C e ajusta
             checa_rpm(20); //verifica se a rotação é 20rpm e ajusta
             Se (T = 50 && RPM = 20) {
                    ativa_timer(); // ativa o timer
                    inicia_timer=1; //flag que o timer está ativo
             }
             Senão (aguarda e repete)
      }
      se (X == 2) { inicia a etapa 2 T=65 e RPM=30
             checa_temperatura(65); //verifica se a temperatura atingiu 65°C e ajusta
             checa_rpm(30); //verifica se a rotação é 30rpm e ajusta
             Se (T = 50 && RPM = 20) {
                    ativa_timer(); // ativa o timer
                    inicia_timer=1; //flag que o timer está ativo
             }
      }
      se (X == 3) { inicia a etapa 3 T=80 e RPM=40
             checa_temperatura(80); //verifica se a temperatura atingiu 80°C e ajusta
             checa_rpm(40); //verifica se a rotação é 40rpm e ajusta
             Se (T = 80 && RPM = 40) {
                    ativa_timer(6; // ativa o timer
                    inicia_timer=1; //flag que o timer está ativo
             }
      }
}

inicia_aquecimento(T_desejada) {
  // Configurar o DAC para gerar uma tensão proporcional a T_desejada
  valor_DAC = converte_temperatura_para_DAC(T_desejada); // Função para converter a
temperatura em valor de DAC
  configura_DAC(valor_DAC); // Função para configurar o DAC com o valor calculado

  // Aguardar a temperatura estabilizar (opcional)
  enquanto (checa_temperatura(T_desejada) != true) {
    delay(100); // Aguarda um curto período de tempo
  }
}
```

```
inicia_rotação(RPM_desejado) {
  // Calcular a frequência do PWM para a velocidade desejada
  frequencia PWM = converte RPM para PWM(RPM desejado); // Função para converter RPM
em frequência de PWM
  configura_PWM_motor(frequencia_PWM); // Função para configurar o PWM do motor com a
frequência calculada

  // Aguardar a velocidade estabilizar (opcional)
  enquanto (checa rpm(RPM desejado) != true) {
    delay(100); // Aguarda um curto período de tempo
  }
}

void checa_temperatura(float T_desejada) {
  // Leitura do sensor de temperatura
  float temperatura = le_ADC(); // Função para ler o valor do ADC
  le_ADC(adc); //ler valor do ADC;
  converte_Temp(temperatura); // Função para converter o valor do ADC para
temperatura em °C


void checa_rpm(int RPM_desejado) { //Verifica se a rotação atingiu o valor desejado
  int RPM = le_rpm(); // Função para ler os pulsos do encoder e calcular a velocidade
em RPM

  // Controle da velocidade do motor
  if (RPM < RPM_desejado) {
    aumentar_PWM_motor(); // Função para aumentar a frequência do PWM do motor
  } else if (RPM > RPM_desejado) {
    diminuir_PWM_motor(); // Função para diminuir a frequência do PWM do motor
  }
}

// Controle da temperatura
controla_temperatura(intensidade) { //Ajusta a saída do controlador de chamas para
mais ou para menos
  if (temperatura < T_desejada) {
    aumentar_PWM_chama(); // Função para aumentar a frequência do PWM do controlador
de chamas
  } else if (temperatura > T_desejada) {
    diminuir_PWM_chama(); // Função para diminuir a frequência do PWM do controlador
de chamas
  }
}

  // Controle da velocidade do motor
    se (velocidade < setpoint) {
      aumentar a frequência do PWM do motor;
    } senão se (velocidade > setpoint) {
      diminuir a frequência do PWM do motor;
    }

// Monitoramento do botão de emergência
checa_botao() {
    se (botão de emergência pressionado) {
      desligar motor;
      desligar chama;
      ativar sirene;
    }
}

// Função que mostra os dados das variáveis no display HID
mostra_display(){
      print "Etapa (X)"; //informa uqal a etapa do porcesso (0,1,2,3 - Finalizado,
Deligado)
      Print "Tempo trascorrido (time)"; //informa o tempo trascorrido de cada etapa
      print "Temperatura = (T)"; // informa a temperatura da caldeira
      print "Rotação = (RPM)"; //informa a rotação do motor
      print "Botão Alerta = (On/Off)"; //mostra o status do botão de alerta
      print "Alarme Sonoro = (On/Off)"; //mostra o status do alarmes
      print "Nível PWM Temperatura = (PWM1)"; //mostra o nível do sinal PWM1
      print "Nível PWM Controlador de Chamas = (PWM2)"; //mostra o nível do sinal
PWM1
}

void checa_timer(int etapa) { //checa o timer para cada uma das etapas
```

```
  unsigned long tempo_decorrido = millis() - tempo_inicial; // Calcula o tempo
decorrido em milissegundos

  switch (etapa) {
    case 1:
      if (tempo_decorrido > 30 * 60 * 1000) { // 30 minutos
        flag_etapa = 2;
        ativa_timer(); // Reinicia o timer para a próxima etapa
      }
      break;

    case 2:
      if (tempo_decorrido > 30 * 60 * 1000) { // 30 minutos
        flag_etapa = 3;
        ativa_timer(); // Reinicia o timer para a próxima etapa
      }
      break;

    case 3:
      if (tempo_decorrido > 60 * 60 * 1000) { // 60 minutos
        flag_etapa = 4; // Fim do processo
      }
      break;
  }
}
void ativa_timer() { //Função que inicializa a comntagem do timer
  tempo_inicial = millis(); // Armazena o tempo atual em milissegundos
}
```

3. Referencias:

[1] - NUCLEO-L476RG: https://www.st.com/en/evaluation-tools/nucleo-l476rg.html#st_all-features_sec-nav-tab

[2] - Motor de Passo HT34-487-R10P: https://loja.kalatec.com.br/motor-de-passo/motor-de-passo-nema-130-kg-com-redutor-planetario-10-1 [

3] - Drive Motor de Passo STR8: https://loja.kalatec.com.br/driver-para-motores-de-passo/str8-drive-8-amperes

[4] - Encoder: Baumer ITD 40 A 4 Y79: https://www.baumer.com/br/pt/resumo-dos-produtos/encoders-e-sensores-de-angulo/encoder-rotativo-industrial-incremental/grande-eixo-oco/eixo-oco-10-ate-27-mm/itd-40-a-4-y79/p/32208

[5] - Sensor de temperatura: Baumer TER-8 de -40°C a 115°C: https://www.baumer.com/br/pt/resumo-dos-produtos/sensores-de-processo/medicao-de-temperatura/higiene/flush/ter8/p/34149

[6] - Sirene TUCSG200: https://www.tucanobrasil.com.br/intru/sir/sire.htm

[7] - Controlador da chama: Kromschröder BCU 460: https://www.kromschroeder.de/marketing/adlatus/techlipedia/out/en/06/BCU_46x_Rev/TI/MAP_TI_BCU_46x_id_map_krs_20200522_135512.html

[8] - Display Industrial 7 polegadas com SPI: https://riverdi.com/product/ritft70?srsltid=AfmBOooP8wRs0VCVMJ4QbcVIzNiH7Rd2BCzCIMlmGn6wOem4B3-1Idm0