CENTRO UNIVERSITÁRIO ENIAC

AUTORES
TYRONNE ALVES DE SOUZA
ORIENTADOR
Prof. Me. Lucas Carlos de Sousa

DESENVOLVIMENTO DE UM SISTEMA DE AUTOMAÇÃO PARA MONITORAMENTO DE VELOCIDADES EM CAIXA DE REDUÇÃO SIMPLES COM SENSORES E MICROCONTROLADOR

Guarulhos
2024

1. RESUMO

Esse projeto tem como finalidade o estudo de como desenvolver uma caixa de redução simples com monitoramento de velocidade através de sensores e microcontrolador, ele contém uma boa parte de programação e eletrônica, as conexões com o ESP 32, vai desde como usar um atuador ou um motor até configurar botão e tela lcd, um projeto completo que vai animar qualquer pessoa a fazer um igual, tenha uma ótima leitura e bons estudos.

2. INTRODUÇÃO

O monitoramento de velocidades em sistemas mecânicos, especialmente em caixas de redução, é um desafio crítico na indústria, impactando diretamente a eficiência, segurança e durabilidade dos equipamentos. Além disso, um dos problemas de acordo com o blog (Dynamox, 28/03/2024), é o desalinhamento, o desalinhamento de uma caixa de redução é uma falha bastante comum em eixos e podem gerar o desgaste anormal dos componentes. Inclusive, pode ocasionar problemas de engrenamento pelo desalinhamento entre as engrenagens.

O sistema de redução de velocidade é muito importante para a sociedade,ocupando várias áreas da indústria. O grande número de aplicações oferecidas inclui, por exemplo: extrusoras, plataformas, rosca alimentadora, paletizadoras, equipamentos de elevação, máquinas dosadoras, misturadores, agitadores, bombas, sistemas de movimentação e esteiras para armazéns e silos (Global Redutores,2017).

Entre os benefícios do uso de um redutor, destacam-se: Redução de custos – o dispositivo melhora a atuação de um motor, o que pode evitar a compra de um novo; Proteção de equipamentos – o redutor ajuda a preservar o maquinário, principalmente em caso de sobrecarga, quando ele é atingido primeiro. Seu custo de reposição é relativamente baixo; Economia de energia – o dispositivo pode reduzir o consumo de energia de sistemas, além de minimizar danos, com o motor funcionando em velocidades menores e evitando a geração de calor excessivo.(Grupo Kalatec, 2024).

Para que tudo ocorra bem é necessário ter uma boa manutenção desses redutores, a manutenção de redutores preventiva e preditiva gera inúmeros benefícios para o processo fabril. Tanto por prolongar a vida útil dos dispositivos quanto por prevenir e

evitar problemas maiores, que podem afetar todo o sistema produtivo industrial, por isso o estudo da manutenção industrial assim como o monitoramento de velocidade desses redutores de velocidade é essencial (Abecom, 2022).

3. OBJETIVOS

Objetivo Geral: Desenvolver um sistema de automação eficaz para monitoramento de velocidades em caixas de redução simples, utilizando sensores e microcontroladores, com a meta de melhorar a segurança operacional, reduzir o consumo de energia e minimizar a geração de resíduos.
Objetivos Específicos: Assim como eu citei no objetivo geral as minhas metas, vou falar como pretende chegar a esses objetivos. Melhorar a Segurança Operacional: Verificar o uso do sensor para ter o monitoramento da temperatura para evitar qualquer tipo de acidente. De acordo com o datasheet do LM35 como exemplo, o custo mais baixo é garantido pelo Corte e calibração no nível do wafer(Texas Instruments, 2017).
Reduzir o Consumo de Energia: verificar a função do microcontrolador ESP32 que possui um modo de economia de energia, chamado “Deep Sleep”. Nesse modo, as CPUs, a maior parte da RAM e todos os periféricos digitais com clock estão desligados. Assim otimizando o uso de energia(Fernando K, 08/2024).

Minimizar a Geração de Resíduos: Verificando a pressão da caixa de redução através do sensor BMP180 como exemplo, pode ajudar a garantir que a caixa de redução opera dentro dos parâmetros ideais. Operar dentro das especificações pode aumentar a eficiência do sistema e reduzir a necessidade de manutenção e substituição, minimizando o impacto ambiental. E de acordo com o datasheet A interface I2C permite fácil integração do sistema com um microcontrolador.(Bosch Sensortec,2013).

4. METODOLOGIA

Esse trabalho é um estudo de caso experimental, com foco em aprender a desenvolver através de um software uma caixa de redução simples com o monitoramento de velocidade, e para isso usaremos alguns itens:

CAIXA FIXADORA: A caixa fixadora terá 15cm de comprimento, 10 de largura e 8 de altura e foi feita em modelo 3d no software autocad.

MOTOR - O motor é um motor de passo chamado Nema 23, tem um diâmetro do eixo de 8mm, torque de 3 Nm uma corrente de 4A, seu drive indicado é o TMA.DMP.556D, tem ângulo de passo de 1,8° e Tem 4 fios que são conectados ao drive para o controle do motor. Usei a fórmula de 360/ângulo de passo para saber os passos de revolução.(Tecmaf, 2024).

DRIVE: o TMA.DMP.556D que é indicado para o motor Nema 23, ele tem:

• micro passo de alta resolução na saída, portanto deixa o movimento mais suave.

• 15 resoluções de micropasso selecionáveis, como: 400, 800, 1600, 3200, 6400, 12800, 25600, 1000, 2000, 4000, 5000, 8000, 10000, 20000, 25000.

• Tensão de alimentação de 24 a 50 VDC.

• 8 valores de corrente de pico selecionáveis, 1.5A, 2.1A, 2.7A, 3.2A, 3.8A, 4.3A, 4.9A, 5.6A.

• Frequência de entrada de pulsos de até 200kHz, TTL compatível e isolado opticamente.

• Redução automática de corrente quando o motor está parado (selecionável pelo DIP4).

• Suporta modo de Pulso e Direção.

• Proteção contra sobrecorrente e sobretensão.

O drive TMA.DMP.556D tem 2 conectores chamados p1 e p2, os 4 fios do motor estão conectados no p2 do drive.(Tecmaf, 2024).

ENGRENAGENS - usaremos 4 engrenagens nessa caixa de redução simples e todas elas tem 8mm de Diâmetro.

• Engrenagem 1 - essa engrenagem vai no eixo do motor, ela tem 5 dentes

• Engrenagem 2 - essa engrenagem vai no eixo 2, essa primeira engrenagem é conectada a engrenagem do eixo do motor e tem 10 dentes. Para o aumento de torque uso a Relação de transmissão que é dada pela fórmula: RT=Z2/Z1 Onde Z1 é o número de dentes da engrenagem motora (do motor) e Z2 é o número de dentes da engrenagem do segundo eixo. Para aumentar o torque e reduzir a velocidade, a engrenagem do motor deve ter menos dentes que a engrenagem do segundo eixo.

• Engrenagem 3 - essa engrenagem está no 2 eixo também, está acoplada a engrenagem 4 do 3 eixo e ela tem 2 dentes.

• Engrenagem 4 - essa engrenagem está no eixo 3 e está acoplada a 2 engrenagem do eixo 2, essa Engrenagem tem 6 dentes.

ATUADOR - D24-450-150LP-IP65 - Atuador Linear 450N - 150mm

Modelo KALATEC: KTC-D24-450-150LP-IP65 (3259), Capacidade de Carga: 450N (45Kg), Alimentação: 24VDC, Corrente: Máximo 2A, Velocidade Máxima sem carga: 20mm/s, Velocidade Máxima com carga: 15mm/s, Potenciômetro interno 10K, Fim de Curso Interno (não ajustável), Temperatura Ambiente : -25oC a +65oC, Regime de Trabalho: Máximo 25%, Grau de Proteção: IP65, Vida útil: 50.000 Ciclos, Material: Aluminio 6062, Curso Útil: 150mm, Cota L1: 292mm, Comprimento Total Estendido: 445mm +/- 1mm, (Kalatec Automação,2024).

MICROCONTROLADOR -

o microcontrolador que irei usar é o esp 32 kit dev v1 possui tensão de operação de 3.3v, entrada de tensão de 7 a 12v, 25 portas digitais, 6 analógicas de entrada e 2 de saída,UARTs: 3, SPIs: 2, I2Cs: 3, Flash Memory: 4 MB, SRAM: 520 KB, Clock Speed: 240 Mhz e Wi-Fi: IEEE 802.11 b/g/n/e/i etc.(Zerynth,2024).

TELA LCD -

a tela lcd vai ser usado tft lcd que se encaixa melhor no projeto. TFT LCD Touchscreen 2.8" com Driver ILI9341 (320x240), Características:, Tamanho: 2,8 polegadas.

Resolução: 320x240 pixels., Tipo de Touch: Resistivo (precisa de um conversor AD para capturar as coordenadas de toque). Driver: ILI9341. Comunicação: Interface SPI. Compatibilidade: Suporta ESP32 via pinos SPI (MISO, MOSI, SCK, CS).(AlienExpress,2024).

SENSORES -

LM35 - é um sensor de temperatura que vai monitorar as engrenagens se estão trabalhando de forma regular sem sobreaquecer, isso vai ajudar a diminuir a manutenção industrial e fazer uma manutenção preventiva para que não ocorram surpresas durante o uso. Usarei 4 sensores LM35. O LM35 tem algumas características que podem ajudar como: calibração direta dos graus centígrados,

opera entre 4v e 30v, menos que 60 Picoamperes e vai de -55° graus a 150° celsius e a cada 10mv é 1° celsius. Usamos a fórmula: Dezenas de V =(Valor lido/1023)×Dezenas de referências. Para calcular a tensão que o sensor leu e dividir pela quantidade que a porta que o Esp 32 pode ler x a referência e então teremos o valor em tensão que depois é dividido pelo valor de mv que o sensor transforma em graus celsius.(National Semiconductor,2000).

A3144 - é um sensor de velocidade que vai gerar pulsos toda vez que um imã passar pelo sensor, ele funciona de 4,5v a 24v, vou usar um resistor de 10k para um controle melhor mas não é necessário.(Allegro MicroSystems,2024).

FONTE DE ALIMENTAÇÃO - LRS-350 Fonte AC/DC 350 Watts Alta Eficiência MeanWell. Para saber qual fonte usar é só somar quanto de amperes consome e qual é a maior tensão necessária, então eu apenas somei.(MW Mean Well,2022).

Esses serão os métodos usados para o desenvolvimento da caixa de redução.

Esses dados foram levantados através de pesquisas na internet como artigos, blogs e sites e um pouco de Chat Gpt. Eu levantei os dados bibliográficos por meio de cada site, artigo e blog que visitei para fazer uma análise e pesquisa desse projeto.

usei as ferramentas MatLab e Autocad. Essas ferramentas foram muito necessárias para o projeto andar pois me auxiliou nas contas e visualização do projeto.

PROGRAMAÇÃO -

Estou utilizando a IDE Arduino IDE para começar a programar o ESP32.

Utilizando a biblioteca TFT (ILI9341) do display.

após isso eu crio um objeto e começo a comunicação serial do display, depois uso o tft.set para configurar o display.

após isso eu configurei o motor de passo, para isso eu adicionei a biblioteca Arduino.h

após isso configurei o atuador.

após isso incluir a biblioteca TFT eSPI.h para a customização da tela

A expressão (5.0 / 1023.0) * 100 utilizada na programação do sensor de temperatura LM35 é uma fórmula que converte a leitura analógica do sensor em uma temperatura em graus Celsius.

5. DESENVOLVIMENTO

MOTOR - primeiro conectamos o motor na caixa fixadora colocando uma haste com dois parafuso para prender o motor a caixa fixadora deixando o eixo do motor para dentro da caixa com uma folga de 1 mm a mais do que o eixo do motor. depois conectamos os 4 fios do motor nos terminais A-, A+,B-,B+ do drive. para encontrar onde estão esses terminais, eles estão no datasheet nos conectores p2 do drive.

ENGRENAGENS - para o cálculo do torque eu fiz a relação de transmissão da 1 marcha, fazendo 10 dentes da 1 engrenagem do 2 eixo dividido pela 1 engrenagem do motor de 5 dentes assim 10/5 dar uma relação de transmissão de 2 por 1 ou seja é

2. então fazemos a multiplicação do torque inicial do motor que é 3Nm x 2 da relação de transmissão que dar 6Nm da 1 marcha. Para o cálculo de torque da 2 marcha eu fiz a relação de transmissão da 2 engrenagem do 2 eixo com 2 dentes e a engrenagem do 3 eixo com 6 dentes. fazendo o cálculo dar 6/2=3 dando uma relação de 3 por 1 de transmissão, então fazendo o toque inicial da 1 marcha que é 6Nm x a relação de transmissão que é 3 dar 18 Nm de torque na 2 marcha.

EIXOS - A engrenagem do motor e a 1 engrenagem do 2 eixo estarão a 5 mm da caixa fixadora acoplados entre si. a 2 engrenagem do eixo 2 mais a engrenagem do eixo 3 estão acopladas a uma distância de 10mm em relação da 1 engrenagem do 2 eixo.

ATUADOR - O atuador tem 2 fios um positivo e o outro negativo, os fios serão conectados ao drive do motor, o negativo será no gnd e o positivo no vcc do drive. ele estará posicionado numa posição ao lado da caixa fixadora, a velocidade do atuador é de que com carga máxima de 45kg ele consegue se mover a 15mm/s, como o peso dos meus eixos + engrenagens estão abaixo desse valor então está tudo bem, a posição das engrenagens estão na metade de 15mm/s para levar metade de 1s para mudar a marcha.

MICROCONTROLADOR

O microcontrolador será conectado ao drive de motor, o PUL+ vai no d17 do ESP 32 para controlar os pulsos de passo. O PUL- ao GND do ESP32. O DIR+ vai no d16 do ESP 32 para definir a direção do motor. O DIR- ao GND do ESP32.

TELA LCD - CONEXÃO E FUNÇÃO

O pino VCC da tela vai para o 3.3v do ESP32. Alimentação

O pino GND da tela vai para o GND do ESP32. Terra

O pino CS da tela vai para o D5 do ESP32. Seleção de chip(CS)

O pino RESET da tela vai para o D4 do ESP32. Reset da Tela

O pino DC/RS da tela vai para o D2 do esp32.data/comando(D/C)

O pino SDI/MOSI da tela vai para o D23 do ESP32. Master Out Slave In

O pino SCK da tela vai para o D18 do ESP32. Clock SPI

O pino LED da tela vai para o 3.3V ou 5V do esp32. Alimentação de fundo.

SENSORES

irei usar 4 sensores LM35. o 1 vai ficar ao lado da 1 engrenagem do motor, o 2 na 1 engrenagem do eixo 2, a 3 na 2 engrenagem do eixo 2 e a 4 no a engrenagem do eixo 3. estou posicionando esses sensores de temperatura para que eu verifique se as

engrenagens estão em temperatura dentro das especificações delas para que não ocorra desbaste ou superaquecimento que faça com que a engrenagem venha a se desfazer, e assim prejudicar o funcionamento da caixa de redução. As conexões estarão no VCC do ESP32 e GND do ESP32, e o último fio ligarei nas portas D15,

D25, D26, D27 do ESP32. Usarei 3 sensores de velocidade A3144 e eles ficarão posicionados 1 na 1 engrenagem do eixo do motor, 2 na 1 engrenagem do eixo 2 e a 3 na engrenagem do eixo 3, colocarei imãs com parafuso nessas 3 engrenagens para que a cada volta que ela der, ela passar para pelo sensor e o sensor gerar pulsos para medir a velocidade das engrenagens. As conexões são 3 fios os vcc no 5v no ESP32. o GND no ESP32 e o outro fio próximo às portas do VCC e porta digital do ESP32. As portas serão D12, D13 e D14.

FONTE DE ALIMENTAÇÃO - LRS-350 Fonte AC/DC 350 Watts Alta Eficiência MeanWell usarei essa fonte para alimentar o meu sistema fazendo a soma de corrente Total estimado de corrente = 4A + 2A + 0,5A + 0,1 A = 6,6A e de tensão é 24v pq é exigido no atuador e do motor. então a ligação fica o terminal positivo vai ao vcc do drive e o negativo no gnd do drive, tbm coloque o positivo na tensão de entrada do Esp32 e no GND do ESP32.

PROGRAMAÇÃO - fazer o download do Arduino IDE para começar a fazer a programação.

após isso eu baixei a biblioteca TFT (ILI9341) para configurar o display

após isso eu configurei o display assim:

```
#include <TFT_eSPI.h> // Inclui a biblioteca para o display TFT
// Criação do objeto TFT
TFT_eSPI tft = TFT SPI(); // Cria um objeto TFT
void setup() {
// Inicializa a comunicação serial
Serial.begin(115200);
// Inicializa o display
tft.init();
tft.setRotation(1); // Define a rotação do display
tft.fillScreen(TFT_BLACK); // Limpa a tela com fundo preto
// Exibe uma mensagem de boas-vindas
tft.setTextColor(TFT_WHITE); // Define a cor do texto
tft.setTextSize(2); // Define o tamanho do texto
```

```
tft.setCursor(10, 10); // Define a posição do cursor
tft.println("Sistema Iniciado!"); // Exibe a mensagem
}
```

criei um objeto TFT e inicializei a comunicação serial com o display, definir a rotação, limpei a

tela e definir cor, tamanho do texto, posição do cursor e por fim exibir uma mensagem.

após isso eu fui para a configuração do motor de passo e fiz esse código:

```
#include <Arduino.h>
// Definições dos pinos
#define PUL_PIN 17 // Pino para pulsos de passo
#define DIR_PIN 16 // Pino para direção
// Variáveis de controle
int stepsPerRevolution = 200; // Número de passos por revolução (ajuste conforme necessário)
int desiredRPM = 60; // RPM desejada
int stepDelay = 1000; // Tempo entre os pulsos em microssegundos
void setup() {
Serial.begin(115200);
pinMode(PUL_PIN, OUTPUT); // Configura o pino de pulsos como saída
pinMode(DIR_PIN, OUTPUT); // Configura o pino de direção como saída
// Define a direção do motor (HIGH ou LOW)
digitalWrite(DIR_PIN, HIGH); // Muda para LOW para inverter a direção
}
void loop() {
// Calcula o intervalo entre os pulsos em microssegundos
int pulseInterval = (60 * 1000000) / (desiredRPM * pulsesPerRevolution);
// Gira o motor em uma direção
for (int i = 0; i < pulsesPerRevolution; i++) {
digitalWrite(PUL_PIN, HIGH); // Pulso alto
delayMicroseconds(10); // Duração do pulso
digitalWrite(PUL_PIN, LOW); // Pulso baixo
delayMicroseconds(pulseInterval - 10); // Intervalo entre pulsos
}
```

```
delay(1000); // Espera 1 segundo antes de mudar a direção
// Inverte a direção do motor
digitalWrite(DIR_PIN, LOW); // Muda a direção
for (int i = 0; i < stepsPerRevolution; i++) {
digitalWrite(PUL_PIN, HIGH);
delayMicroseconds(stepDelay);
digitalWrite(PUL_PIN, LOW);
delayMicroseconds(stepDelay);
}
delay(1000); // Espera 1 segundo antes de repetir
}
void loop() {
// Calcula o intervalo entre os pulsos em microssegundos
int pulseInterval = (60 * 1000000) / (desiredRPM * pulsesPerRevolution);
// Envia pulsos para o motor
for (int i = 0; i < pulsesPerRevolution; i++) {
digitalWrite(PUL_PIN, HIGH); // Pulso alto
delayMicroseconds(10); // Duração do pulso
digitalWrite(PUL_PIN, LOW); // Pulso baixo
delayMicroseconds(pulseInterval - 10); // Intervalo entre pulsos
}
delay(1000); // Espera um segundo antes de repetir
}
```

Eu incluí a biblioteca Arduino.h e defini a porta 17 e 16 do ESP32. Depois criei uma variável do tipo inteira com o valor de 200 que são os passos de revolução. A conta que eu fiz foi 360/1,8=200. como descrito no método. Criei uma variável do tipo inteira para o delay de 1 segundo. Coloquei os pinos de saída do ESP 32 na configuração. definir a porta DIR com valor no alto para dar movimento no sentido horário. Então no loop eu criei uma variável do tipo inteiro com valor 0 e que seria incrementado sempre eu o pino PUL enviar um pulso alto. Coloquei um delay e pedi pra ler o pino PUL baixo e ler o pulso e depois pedir um delay de novo. depois inverti a direção do motor. também definir a frequência em que o motor irá atuar.

Depois eu configurei o atuador assim:

```
#include <Arduino.h>
```

```
// Definições dos pinos
#define ACTUATOR PIN 23 // Pino para controle do atuador (pode ser ajustado conforme
necessário)
// Variáveis de controle
int actuatorPosition = 0; // Posição inicial do atuador
const int maxPosition = 150; // Posição máxima em mm (ajuste conforme necessário)
int actuatorDelay = 500; // Tempo para o atuador mover (ajuste conforme necessário)
void setup() {
Serial.begin(115200);
pinMode(ACTUATOR_PIN, OUTPUT); // Configura o pino do atuador como saída
// Inicializa o atuador na posição inicial
digitalWrite(ACTUATOR_PIN, HIGH); // Ativa o atuador
delay(1000); // Espera 1 segundo para garantir que o atuador esteja em movimento
digitalWrite(ACTUATOR_PIN, LOW); // Desativa o atuador
}
void loop() {
// Move o atuador para frente
Serial.println("Movendo para frente...");
digitalWrite(ACTUATOR_PIN, HIGH); // Ativa o atuador
delay(actuatorDelay); // Aguarda conforme necessário para a mudança de marcha.
digitalWrite(ACTUATOR_PIN, LOW); // Desativa o atuador
// Gira o motor em uma direção
for (int i = 0; i < stepsPerRevolution; i++) {
digitalWrite(PUL_PIN, HIGH); // Envia pulso alto
delayMicroseconds(stepDelay); // Espera um tempo definido
digitalWrite(PUL_PIN, LOW); // Envia pulso baixo
delayMicroseconds(stepDelay); // Espera um tempo definido
}
```

```
delay(1000); // Espera 1 segundo antes de mudar a direção
// Inverte a direção do motor e move novamente
digitalWrite(DIR_PIN, LOW); // Muda a direção
Serial.println("Mudando marcha...");
digitalWrite(ACTUATOR_PIN, HIGH); // Ativa o atuador
delay(actuatorDelay); // Aguarda o tempo necessário para a mudança de marcha
digitalWrite(ACTUATOR_PIN, LOW); // Desativa o atuador
for (int i = 0; i < stepsPerRevolution; i++) {
digitalWrite(PUL_PIN, HIGH);
delayMicroseconds(stepDelay);
digitalWrite(PUL_PIN, LOW);
delayMicroseconds(stepDelay);
}
delay(1000); // Espera 1 segundo antes de repetir
}
```

eu defini o pino 13 do ESP 32 para o controle do atuador.

criei uma variável do tipo inteira e atribuir o valor 0 como a posição inicial do atuador. Depois criei uma variável do tipo constante com o valor máximo que ela pode assumir que é 150mm como descrito no cursor no método. criei uma variável do tipo inteira com o valor de 500 para dar o tempo do atuador se posicionar. No setup eu coloquei o pino do atuador em baixo que significa que ele está desativado. No loop eu definir o movimento do atuador, eu pedir para imprimir na tela mudando para a 2 marcha quando o atuador estiver se movendo para frente, ele vai ler e se o pino estiver no alto ele ativa o atuador, espera 2 segundos e ajustar conforme necessário e em seguida desativa. e espera 1 segundo e faz movimento contrário, imprimindo a mensagem mudando para a 1 marcha e espera 1 segundo.

configurando a tela e o atuador:

```
#include <TFT_eSPI.h> // Biblioteca para o display TFT
#include <Arduino.h>
// Definições dos pinos
#define ACTUATOR_PIN 23 // Pino para controle do atuador
#define PUL_PIN 17 // Pino para pulsos do motor
#define DIR_PIN 16 // Pino para direção do motor
```

```
// Definições dos sensores
#define LM35_SENSOR_1 A0 // Sensor de temperatura 1
#define LM35_SENSOR_2 A1 // Sensor de temperatura 2
#define LM35_SENSOR_3 A3 // Sensor de temperatura 3
#define LM35_SENSOR_4 A4 // Sensor de temperatura 4
#define A3144_SENSOR_1 12 // Sensor de velocidade 1
#define A3144_SENSOR_2 13 // Sensor de velocidade 2
#define A3144_SENSOR_3 14 // Sensor de velocidade 3
// Variáveis de controle
volatile int pulseCount1 = 0;
volatile int pulseCount2 = 0;
volatile int pulseCount3 = 0;
TFT_eSPI tft = TFT_eSPI(); // Criação do objeto TFT
void setup() {
Serial.begin(115200);
pinMode(ACTUATOR_PIN, OUTPUT); // Configura o pino do atuador como saída
pinMode(PUL_PIN, OUTPUT); // Configura o pino de pulsos como saída
pinMode(DIR_PIN, OUTPUT); // Configura o pino de direção como saída
pinMode(LM35_SENSOR_1, INPUT);
pinMode(LM35_SENSOR_2, INPUT);
pinMode(LM35_SENSOR_3 INPUT);
pinMode(LM35_SENSOR_4 INPUT);
pinMode(A3144_SENSOR_1, INPUT_PULLUP);
pinMode(A3144_SENSOR_2, INPUT_PULLUP);
pinMode(A3144_SENSOR_3, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(A3144_SENSOR_1), countPulse1, RISING);
attachInterrupt(digitalPinToInterrupt(A3144_SENSOR_2), countPulse2, RISING);
attachInterrupt(digitalPinToInterrupt(A3144_SENSOR_3), countPulse3, RISING);
tft.init();
tft.setRotation(1); // Define a rotação do display
```

```
tft.fillScreen(TFT_BLACK); // Limpa a tela com fundo preto
drawButton();
}
void loop() {
float temp1 = analogRead(LM35_SENSOR_1) * (5.0 / 1023.0) * 100; // Leitura
da temperatura
em Celsius
float temp2 = analogRead(LM35_SENSOR_2) * (5.0 / 1023.0) * 100;
float temp3 = analogRead(LM35_SENSOR_1) * (5.0 / 1023.0) * 100;
float temp4 = analogRead(LM35_SENSOR_1) * (5.0 / 1023.0) * 100;
int speed1 = pulseCount1; // Contagem de pulsos do sensor de velocidade 1
int speed2 = pulseCount2; // Contagem de pulsos do sensor de velocidade 2
int speed3 = pulseCount3; // Contagem de pulsos do sensor de velocidade 3
displayData(temp1, temp2, temp3, temp4, temp2, speed1, speed2, speed3);
delay(1000); // Atualiza a cada segundo
// Reseta as contagens após a leitura
pulseCount1 = pulseCount2 = pulseCount3 = 0;
}
// Função para desenhar o botão na tela
void drawButton() {
tft.fillRect(50, 180, 220, 40, TFT_BLUE); // Desenha o botão
tft.setTextColor(TFT_WHITE);
tft.setTextSize(2);
tft.setCursor(90, 190);
tft.println("Mudar Marcha");
}
// Função para exibir dados na tela
void displayData(float temp1, float temp2, int speed1, int speed2, int speed3) {
tft.fillRect(0, 0, 320, 240, TFT_BLACK); // Limpa a tela
tft.setTextColor(TFT_WHITE);
tft.setTextSize(2);
tft.setCursor(10, 10);
tft.printf("Temp Engrenagem 1: %.2f C\n", temp1);
tft.setCursor(10, 40);
```

```
tft.printf("Temp Engrenagem 2: %.2f C\n", temp2);
tft.setCursor(10, 70);
tft.printf("Velocidade Sensor 1: %d\n", speed1);
tft.setCursor(10, 100);
tft.printf("Velocidade Sensor 2: %d\n", speed2);
tft.setCursor(10, 130);
tft.printf("Velocidade Sensor 3: %d\n", speed3);
}
// Funções para contar pulsos dos sensores de velocidade
void countPulse1() {
pulseCount1++;
}
void countPulse2() {
pulseCount2++;
}
void countPulse3() {
pulseCount3++;
}
```

Eu defini os sensores de temperatura e de velocidade, depois criei variáveis do tipo inteira para contagem de pulso e atribuir o valor de 0 aos 3 sensores de velocidade. ativei a entrada de todos os sensores.

depois eu selecionei que a cada volta que o sensor der que gere um pulso nas portas digitais. no loop eu criei variáveis do tipo float para ler a temperatura dos sensores. irão ler um valor padrão que deve ser até 5.0 e dividir por 1023 e depois multiplicar por 100 para dar o valor em tensão e depois dividir por 10mv que é 0,01 então saberá quantos graus celsius vai ter. Depois criei 3 inteiros para a contagem de pulsos dos sensores de velocidade. depois pedir para mostrar no display todos os valores dos sensores. pedir um delay para atualizar a cada segundo. Após isso a temperatura reseta se os valores são 0. Depois criei uma função para a tela e criei um botão. depois criei outra função para mostrar os valores na tela Depois criei uma função para a contagem de pulsos de cada sensor de velocidade. agora eu finalizo equalizando a velocidade e automatizando a troca de marcha ao atingir a velocidade máxima da 1 marcha e indo para a 2 marcha e depois voltando para a 1a marcha.

```
#include <TFT_eSPI.h>
```

```
#include <Arduino.h>
// Definições dos pinos
#define PUL_PIN 17 // Pino para pulsos do motor
#define DIR_PIN 16 // Pino para direção do motor
#define ACTUATOR_PIN 23 // Pino para controle do atuador
#define LM35_SENSOR_1 A0 // Sensor de temperatura 1
#define LM35_SENSOR_2 A1 // Sensor de temperatura 2
#define A3144_SENSOR_1 12 // Sensor de velocidade 1
#define A3144_SENSOR_2 13 // Sensor de velocidade 2
#define A3144_SENSOR_3 14 // Sensor de velocidade 3
// Variáveis globais
volatile int pulseCount1 = 0;
volatile int pulseCount2 = 0;
volatile int pulseCount3 = 0;
const int pulsesPerRevolution = 200; // Para Nema 23 com ângulo de 1.8°
int desiredRPM = 60; // RPM desejada para a marcha 1
float maxSpeedFirstGear = 100; // Velocidade máxima da marcha 1 em RPM
float maxSpeedSecondGear = 150; // Velocidade máxima da marcha 2 em RPM
void setup() {
Serial.begin(115200);
pinMode(PUL_PIN, OUTPUT);
pinMode(DIR_PIN, OUTPUT);
pinMode(ACTUATOR_PIN, OUTPUT);
pinMode(A3144_SENSOR_1, INPUT_PULLUP);
pinMode(A3144_SENSOR_2, INPUT_PULLUP);
pinMode(A3144_SENSOR_3, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(A3144_SENSOR_1), countPulse1,
RISING);
attachInterrupt(digitalPinToInterrupt(A3144_SENSOR_2), countPulse2,
RISING);
attachInterrupt(digitalPinToInterrupt(A3144_SENSOR_3), countPulse3,
RISING);
digitalWrite(DIR_PIN, HIGH); // Define a direção inicial do motor
}
```

```
void loop() {
static unsigned long lastTime = 0;
unsigned long currentTime = millis();
if (currentTime - lastTime >= 1000) { // A cada segundo
float speed1 = (pulseCount1 / (float)pulsesPerRevolution) * 60; // RPM para o primeiro
sensor
float speed2 = (pulseCount2 / (float)pulsesPerRevolution) * 60; // RPM para o segundo
sensor
float speed3 = (pulseCount3 / (float)pulsesPerRevolution) * 60; // RPM para o terceiro
sensor
Serial.print("Velocidade Sensor 1: ");
Serial.print(speed1);
Serial.println(" RPM");
Serial.print("Velocidade Sensor 2: ");
Serial.print(speed2);
Serial.println(" RPM");
Serial.print("Velocidade Sensor 3: ");
Serial.print(speed3);
Serial.println(" RPM");
pulseCount1 = pulseCount2 = pulseCount3 = 0; // Reseta os contadores
// Verifica se deve mudar de marcha
if (speed1 >= maxSpeedFirstGear) {
changeGear(2); // Muda para a segunda marcha se a velocidade máxima for atingida
} else if (speed1 < maxSpeedFirstGear && speed1 > (maxSpeedFirstGear * 0.9)) {
changeGear(1); // Retorna para a primeira marcha se estiver abaixo da velocidade
máxima
}
lastTime = currentTime;
```

```
}
controlMotor(); // Controla a velocidade do motor baseado na RPM desejada
}
// Funções para contar pulsos dos sensores de velocidade
void countPulse1() {
pulseCount1++;
}
void countPulse2() {
pulseCount2++;
}
void countPulse3() {
pulseCount3++;
}
// Função para mudar a marcha
void changeGear(int gear) {
digitalWrite(ACTUATOR_PIN, HIGH); // Ativa o atuador para mudar marcha
delay(1000); // Tempo necessário para a troca de marcha
digitalWrite(ACTUATOR_PIN, LOW); // Desativa o atuador
if (gear == 2) {
desiredRPM = maxSpeedSecondGear; // Altera a RPM desejada para a segunda marcha
Serial.println("Mudando para segunda marcha.");
} else {
desiredRPM = maxSpeedFirstGear; // Altera a RPM desejada para a primeira marcha
Serial.println("Retornando para primeira marcha.");
}
}
// Função para controlar a velocidade do motor com base na RPM desejada
void controlMotor() {
int pulseInterval = (60 * 1000000) / (desiredRPM * pulsesPerRevolution);
for (int i = 0; i < pulsesPerRevolution; i++) {
digitalWrite(PUL_PIN, HIGH); // Pulso alto
delayMicroseconds(10); // Duração do pulso
```

```
digitalWrite(PUL_PIN, LOW); // Pulso baixo
delayMicroseconds(pulseInterval - 10); // Intervalo entre pulsos
}
}.
```

e assim finalizo meu trabalho que através de muita pesquisa eu conseguir chegar a completar o projeto.

6. RESULTADOS PRELIMINARES

o resultado esperado e que ao ligar a fonte , a tela ligue, e todos os outros componentes liguem, então, eu aciono o motor e mostrará a velocidade que os sensores estão indicando e a temperatura, e quando a velocidade da 1 marcha atingir seu limite, o usuário aperta o botão as engrenagens se ajustam na velocidade e então atuador é acionado para mudar a marcha e na tela deve mostrar mudando para a 2 marcha e quando atingir uma determinada velocidade da 2 marcha que também mostrará na tela, então o atuador liga e volta para a primeira marcha de novo.

7. FONTES CONSULTADAS

Dynamox. Redutor de velocidade: Principais tipos e possíveis falhas. tipo: Blog. Autor: Dynamox. Disponível no site:Redutor de velocidade: principais tipos e possíveis falhas (dynamox.net). acesso em 28 de março de 2024.

Global Redutores. Aplicações dos Redutores na Indústria. Tipo: Blog. Autor: Global Redutores. Disponível no site:Global Redutores - Aplicações dos Redutores na Indústria. acesso em 10 Outubro 2017.

Grupo Kalatec. Redutor de velocidade: entenda a importância para os maquinários industriais. tipo: Blog. Autor: Edilson cravo. Disponível no site: Importância do Redutor de velocidade para as máquinas industriais (kalatec.com.br). acesso em 2024.

Abecom. Manutenção em redutores de velocidade: o que você precisa saber?. Tipo: Site. Autor: Abecom Rolamentos e Produtos de Borracha LTDA. Disponível no

Site: Manutenção em redutores de velocidade: o que você precisa saber? (abecom.com.br). acesso em 06/10/2022.

texas instruments. LM35 Precision Centigrade Temperature Sensors. Tipo: Documento técnico. Autor: Texas Instruments Incorporated. Disponível no Link:LM35 Precision Centigrade Temperature Sensors datasheet (Rev. H). revisado em dezembro de 2017.

Fernando K. Economizando bateria com Deep Sleep!. Tipo: Site. Autor Fernando K. Disponível no Site: Economizando bateria com Deep Sleep! - Fernando K Tecnologia. acesso em 3 de julho de 2024.

Bosch Sensortec. BMP180 Digital pressure sensor. Tipo: Documento técnico. Autor: Bosch Sensortec. Disponível no link: Lorem ipsum | Dolor sit amet (adafruit.com). acesso em 5 de abril de 2013.

Tecmaf. Motor de Passo Nema 23 - 30kgf.cm.Tipo: site. Autor:Tecmaf. Disponível no link:Motor de Passo Nema 23 - 30kgf.cm - Loja TECMAF. acesso em 2024.

Tecmaf. Driver para Motor de Passo - TMA.DMP.556D. Tipo: DataSheet. Autor:Tecmaf. Disponível no link:Microsoft Word - TMA.DMP.556D (tecmaf.com.br). acesso em 2024.

Kalatec Automação. D24-450-150LP-IP65 - Atuador Linear 450N - 150mm. Tipo: Site. Autor:Kalatec Automação. Disponível no link: Kalatec Automação - Atuador linear 450N - 150mm. acesso em 2024.

Zerynth. D24-450-150LP-IP65 - DOIT Esp32 DevKit v1, Tipo: DataSheet. Autor: Zerynth Disponível no link:roboeq.ir/files/id/4034/name/ESP32 MODULE.pdf/. acesso em 2024.

AlienExpress. Módulo de exibição de toque colorido, SPI TFT LCD, ILI9341, ILI9488, 480x320, movimentação 240x320 Tipo: Loja Autor: AlienExpress Disponível no link: Módulo de exibição de toque colorido, SPI TFT LCD, 1.44, 1.8, 2.0, 2.2, 2.4, 2.8, 3.2,

3.5, 4.0 ", ILI9341, ILI9488, 480x320, movimentação 240x320 - AliExpress 502. acesso em 2024.

National Semiconductor. Lm35 Tipo: DataSheet. Autor: National Semiconductor. Disponível no link: LM35 datasheet(1/13 Pages) NSC | Precision Centigrade Temperature Sensors (alldatasheetpt.com). novembro de 2000.

Allegro MicroSystems.Folha de dados do A3144. Tipo: DataSheet. Autor: Allegro MicroSystems.Disponível no link:Folha de dados A3144 (1/8 páginas) ALLEGRO | INTERRUPTORES SENSÍVEIS DE EFEITO HALL PARA OPERAÇÃO EM ALTA TEMPERATURA (alldatasheet.com). Acesso em 2024.

MW Mean Well.LRS-350 Fonte AC/DC 350 Watts Alta Eficiência MeanWell. Tipo: DataSheet. Autor: MeanWell. Disponível no link:未命名 -1 (lojadasfontes.com). Acesso em 2022-08-05