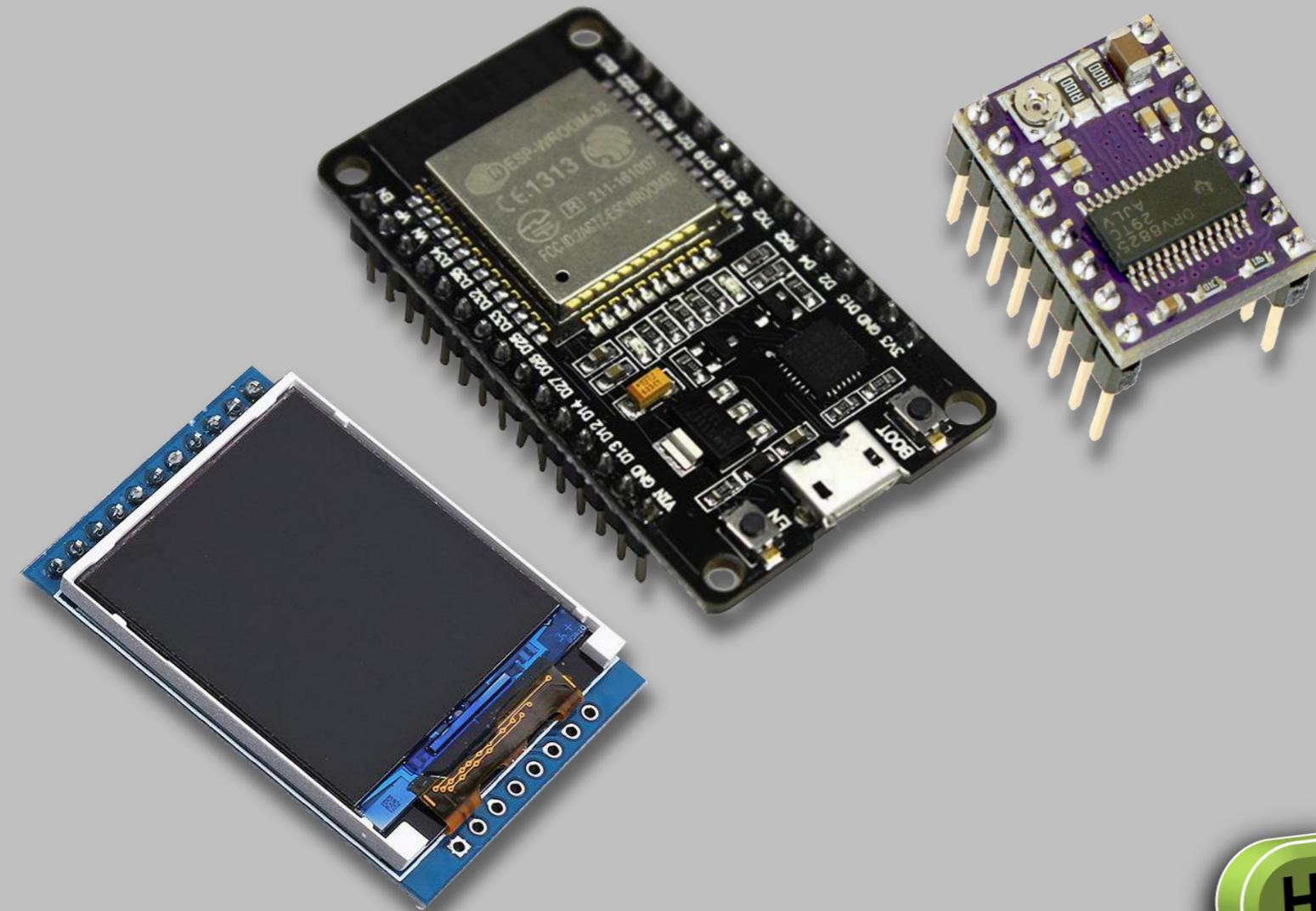


# Laboratório de Motor de Passo



Por Fernando Koyanagi





Em [www.fernandok.com](http://www.fernandok.com)

- PRINCIPAL
- SOBRE FERNANDO K
- ARDUINO
- ESP8266
- ESP32
- LORAWAN
- MOTOR
- DISPLAY
- MATERIAIS
- DOWNLOAD

Receba o meu conteúdo GRATUITAMENTE

QUERO RECEBER GRÁTIS



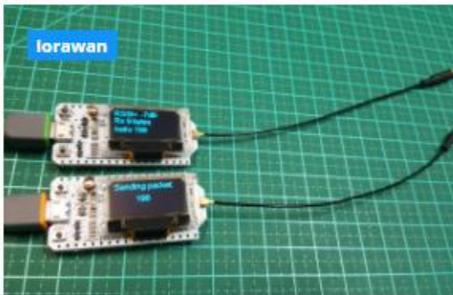
Seu e-mail



### Motor de Passo Nema 23 com Driver TB6600 e Arduino Due

by Fernando K Tecnologia - 2:44 PM  
Hoje vamos voltar a falar de Motor de Passo. Vamos utilizar um Nema 23 que será controlado por um Driver TB6600 e um Arduino Due. É p...

Leia mais



### ESP32 Longa Distância - LoRaWan

by Fernando K Tecnologia - 9:46 AM  
Neste artigo vamos tratar da LoRaWAN, uma rede que vai longe gastando pouca energia. Mas, o quanto "longe"? Com o chip que uso no vídeo...

Leia mais



### Motor de HD com Arduino

by Fernando K Tecnologia - 2:00 PM

#### QUAL ASSUNTO VOCÊ TEM

- Arduino
- ESP8266
- ESP32
- Motor
- Display
- Sensor

You may select multiple answers.  
Votar Exibir resultados

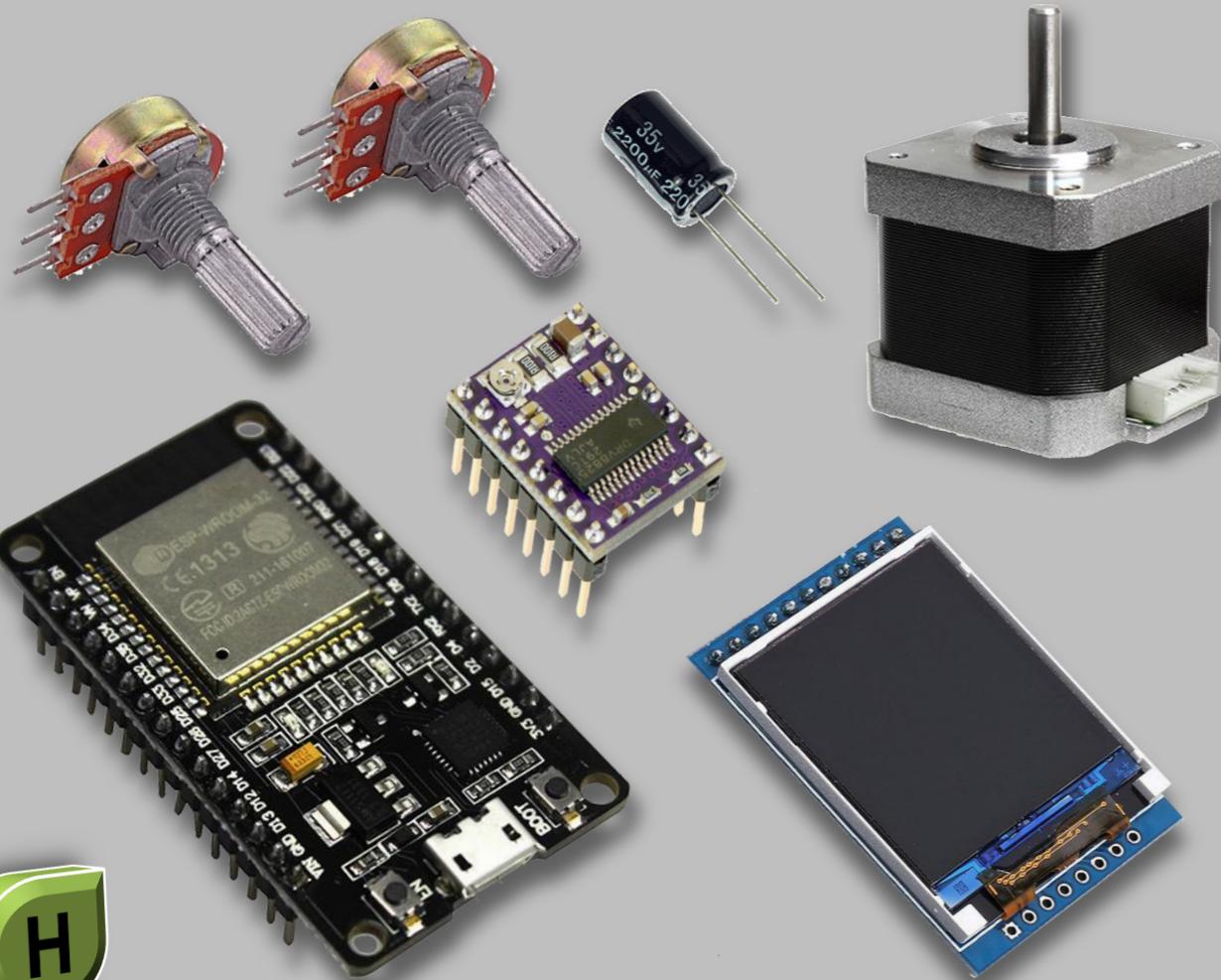
Votos até o momento: 32  
Dias restantes para votar: 49

#### FACEBOOK



# Recursos usados

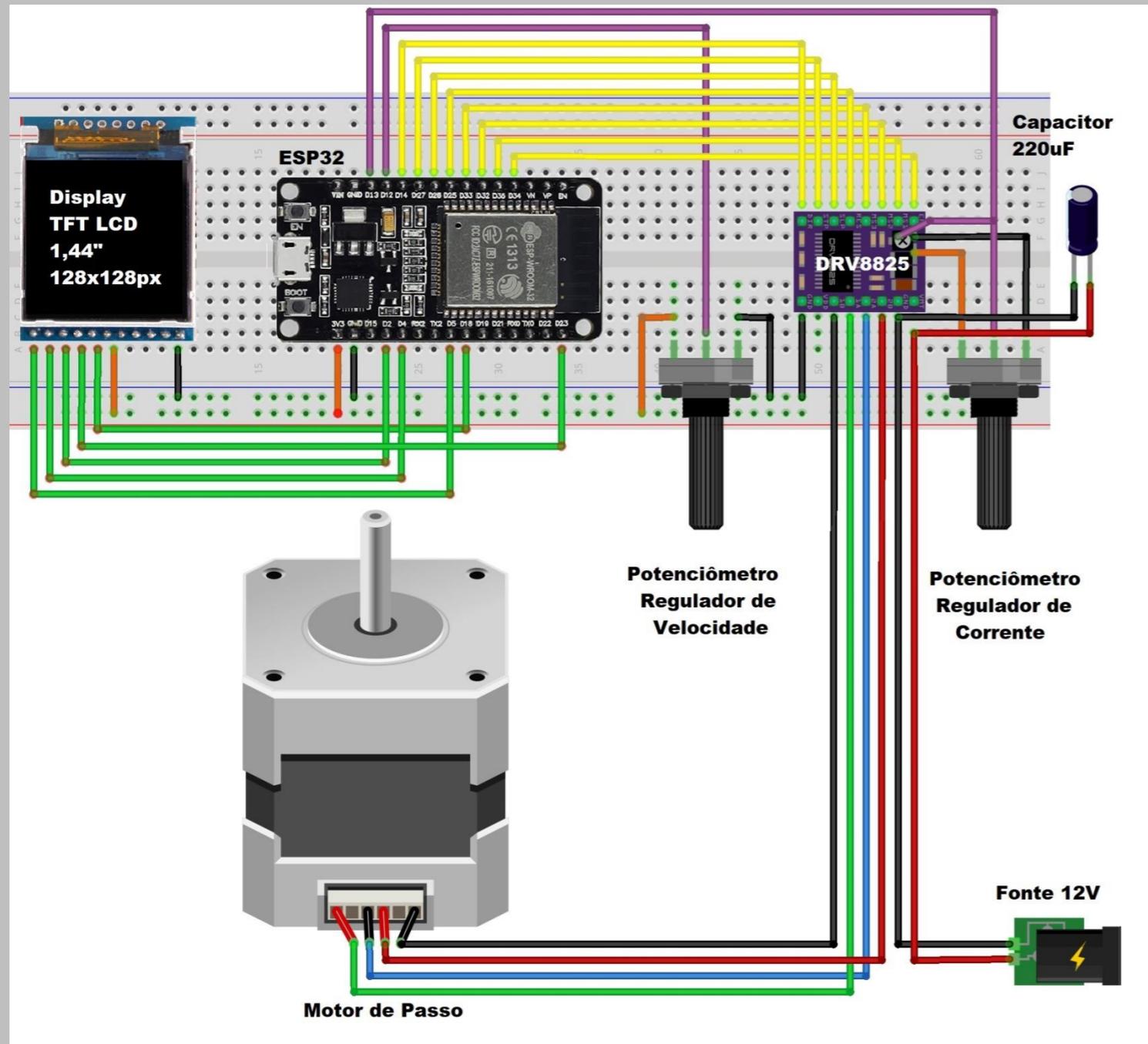
- **ESP WROOM 32**
- **Módulo TFT LCD 1,44" RGB**
- **Driver DRV8825**
- **2 Potenciômetros: 10k e 50k**
- **Capacitor Eletrolítico 220uF**
- **Motor de Passo**



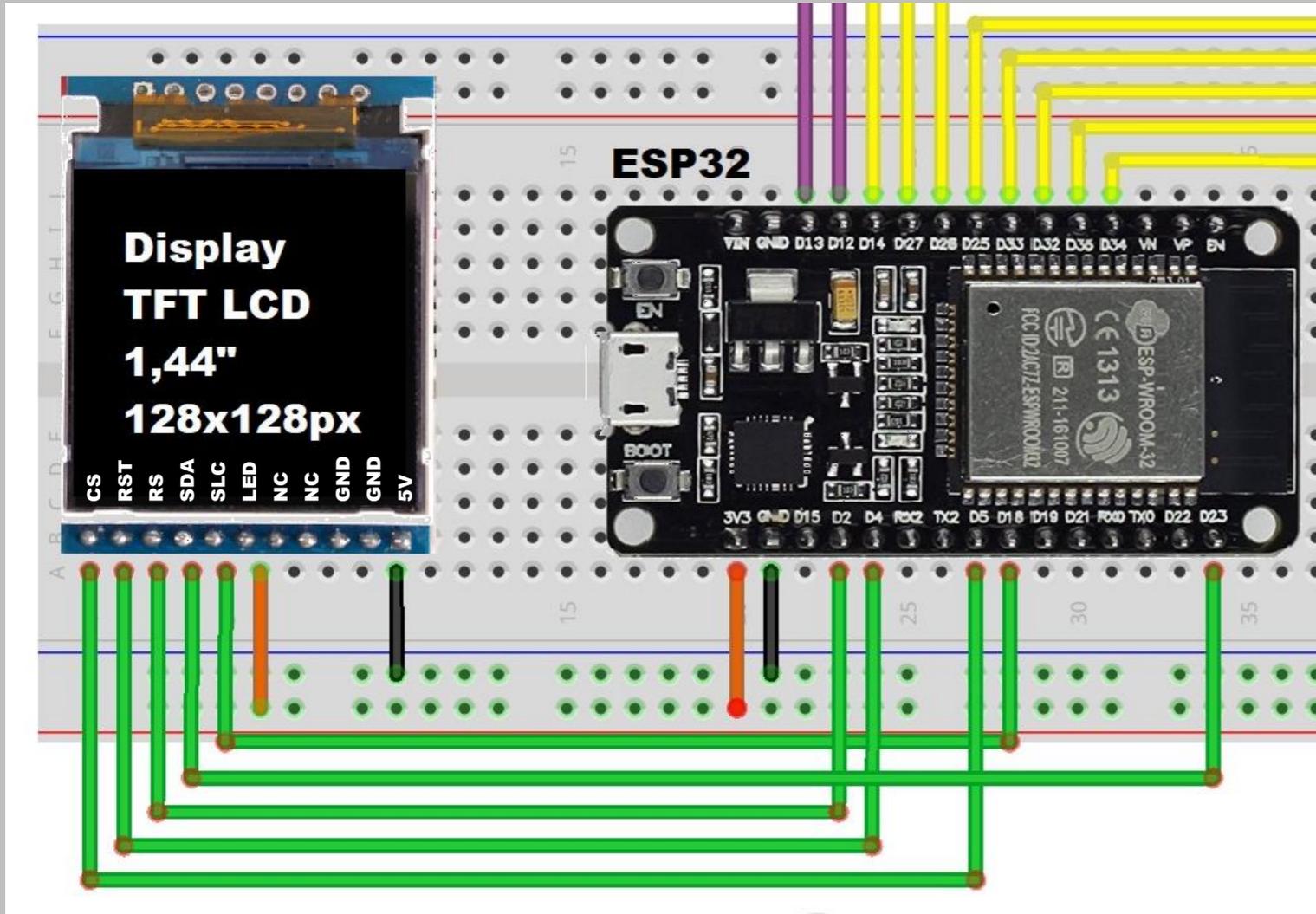
# **Intenção dessa aula**

- 1. Montar um laboratório para motores de passo**
  - 2. Introdução à programação Multitask no ESP 32**
  - 3. Mostrar como calcular valores de tensão, corrente de motores de passo**
  - 4. Testar diversos tipos de motores**
- 
- A hand is shown in a dark, blue-toned environment, pointing towards a glowing digital interface. The interface features a grid of lines and a circular pattern, suggesting a technical or data-driven context. The lighting is dramatic, with strong highlights and deep shadows, creating a futuristic and focused atmosphere.

# Montagem...



# Montagem do Display...

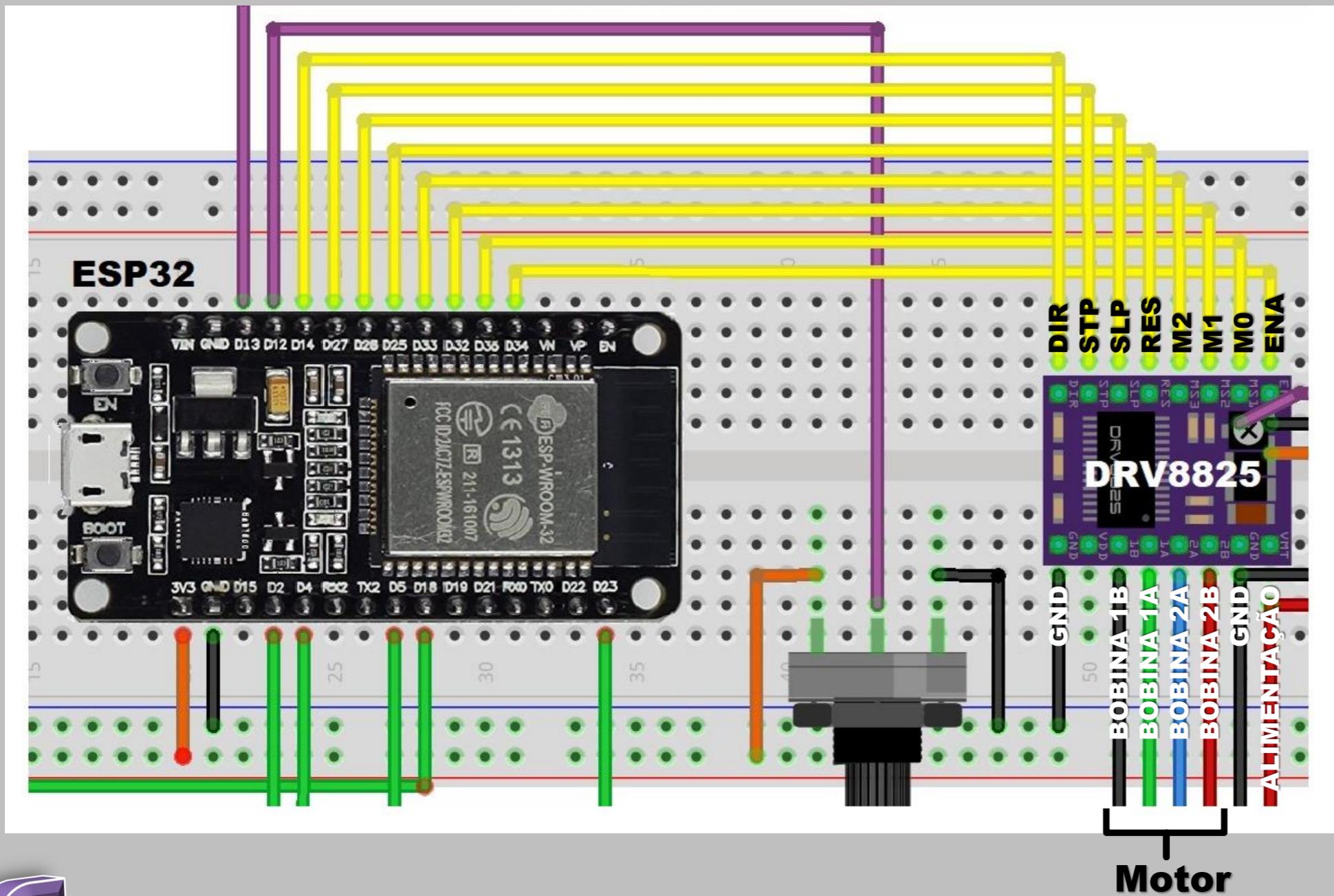


# Ligações do Display...

<b>ESP32</b>	<b>DISPLAY</b>
<b>3V3</b>	<b>5V</b>
<b>GND</b>	<b>GND</b>
<b>3V3</b>	<b>LED</b>
<b>D18</b>	<b>SCL</b>
<b>D23</b>	<b>SDA</b>
<b>D02</b>	<b>RS</b>
<b>D04</b>	<b>RST</b>
<b>D05</b>	<b>CS</b>



# Montagem do DRV8825...



# Ligações do DRV8825...

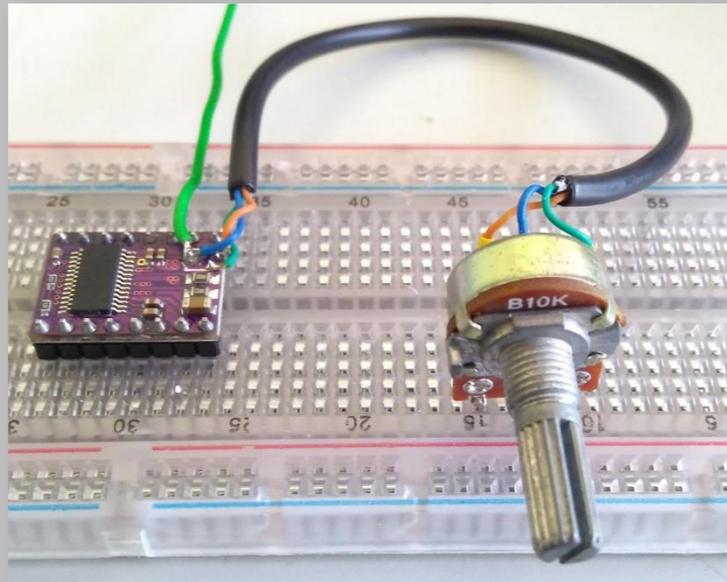
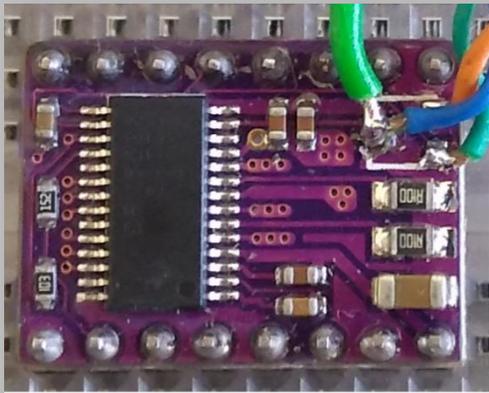
<b>ESP32</b>	<b>DRV8825</b>
<b>D25</b>	<b>RST</b>
<b>D26</b>	<b>SLP</b>
<b>D34</b>	<b>ENA</b>
<b>D35</b>	<b>M0</b>
<b>D32</b>	<b>M1</b>
<b>D33</b>	<b>M2</b>
<b>D14</b>	<b>DIR</b>
<b>D27</b>	<b>STP</b>
<b>GND</b>	<b>GND</b>



# Ligação dos Potenciômetros

	<b>DRV8825 (Controle de Corrente)</b>	<b>VELOCIDADE</b>
<b>ESP32</b>	<b>D13</b>	<b>D12</b>

**Para melhorar a manipulação do controle de corrente do DRV8825, trocamos o potenciômetro de ajuste do driver por outro maior, porém de mesma resistência (10K ohms).**



# Como calibrar a entrada de dados

Para fazer a **calibração** dos dados coletados para calcular a **tensão** sobre o cursor do **potenciômetro** do driver **DRV8825**, usamos o **Excel**.

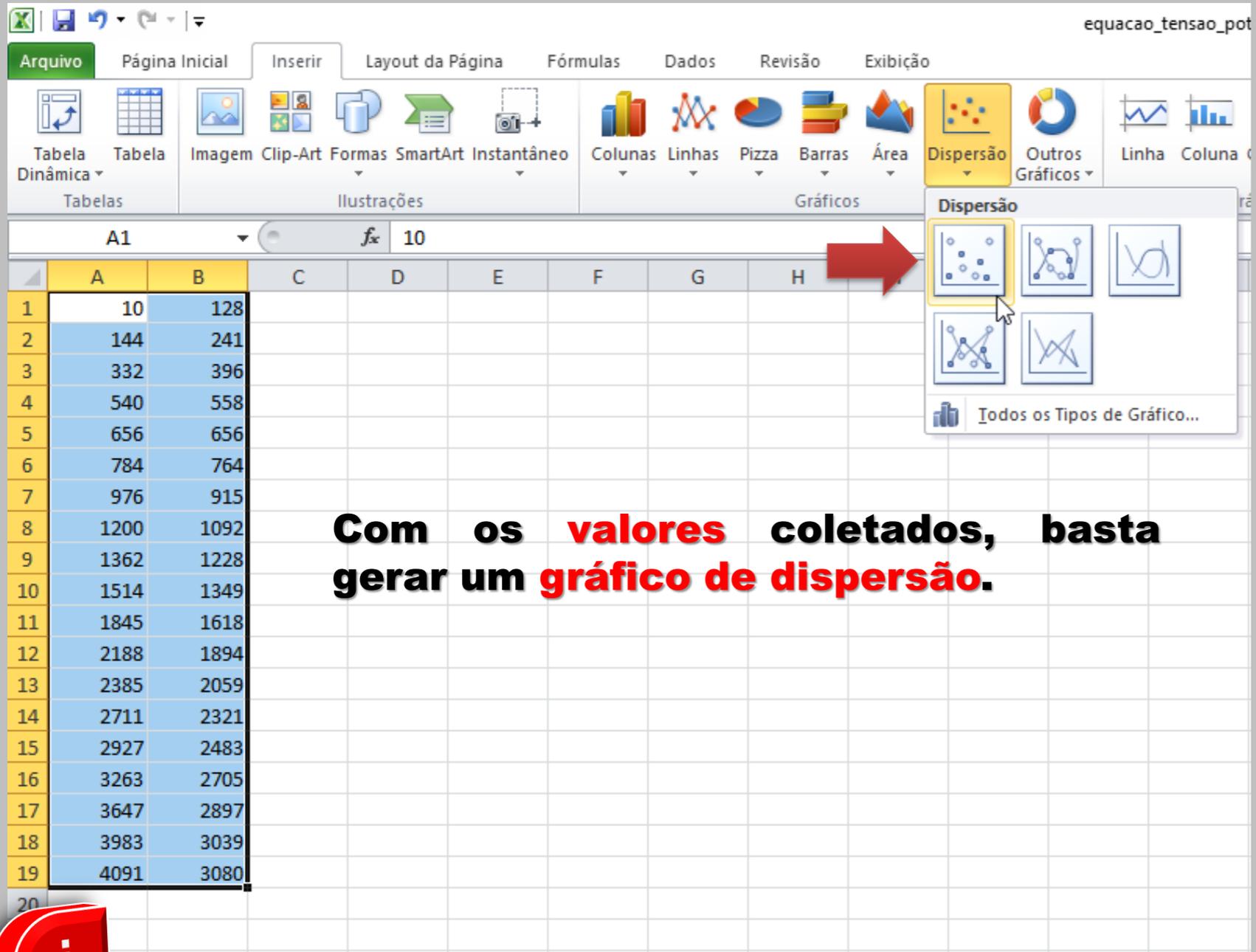
Primeiro **coletamos** os **valores de AD** da porta de entrada do **ESP** que varia de **0 a 4095**.

Com um **multímetro**, medimos a **tensão** do cursor do **potenciômetro** do driver em cada momento.

Valor do AD	Tensão em Milivolts
10	128
144	241
332	396
540	558
656	656
784	764
976	915
1200	1092
1362	1228
1514	1349
1845	1618
2188	1894
2385	2059
2711	2321
2927	2483
3263	2705
3647	2897
3983	3039
4091	3080



# Como calibrar a entrada de dados



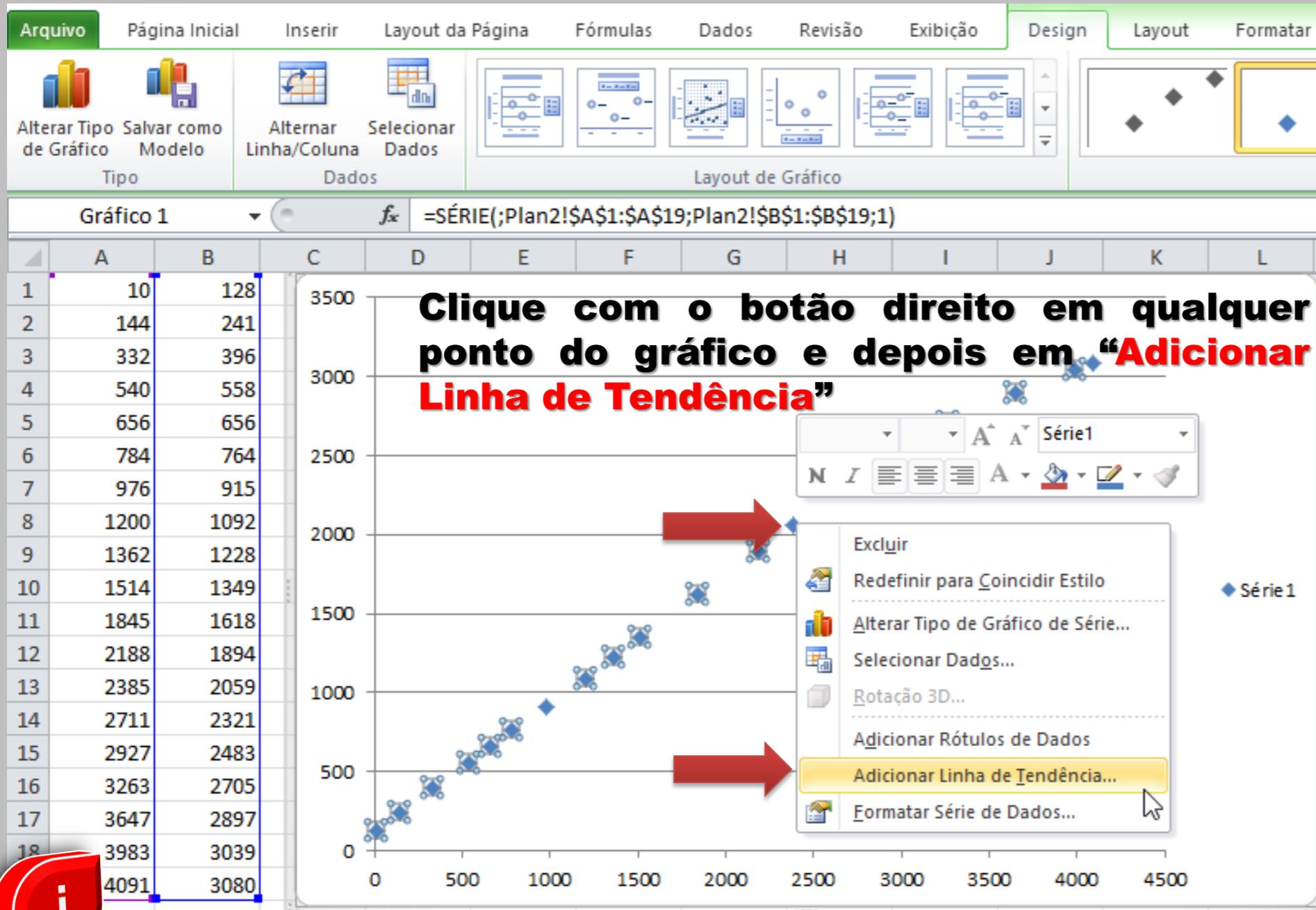
The screenshot shows the Microsoft Excel interface with the 'Inserir' (Insert) tab selected. The 'Gráficos' (Charts) group is active, and the 'Dispersão' (Scatter) dropdown menu is open. A red arrow points from the data table to the 'Dispersão' option in the menu. The data table contains the following values:

	A	B
1	10	128
2	144	241
3	332	396
4	540	558
5	656	656
6	784	764
7	976	915
8	1200	1092
9	1362	1228
10	1514	1349
11	1845	1618
12	2188	1894
13	2385	2059
14	2711	2321
15	2927	2483
16	3263	2705
17	3647	2897
18	3983	3039
19	4091	3080

**Com os valores coletados, basta gerar um gráfico de dispersão.**



# Como calibrar a entrada de dados



# Como calibrar a entrada de dados

Depois selecione "Linear", "Exibir Equação no gráfico" e "Exibir valor de R-quadrado no gráfico".

**Formatar Linha de Tendência**

**Opções de Linha de Tendência**

Cor da Linha

Estilo da Linha

Sombra

Bordas Suaves e Brilhantes

**Opções de Linha de Tendência**

Tipo de Tendência/Regressão

- Exponencial
- Linear
- Logarítmica
- Polinomial Ordem: 2
- Potência
- Média Móvel Período: 2

Nome da Linha de Tendência

- Automático: Linear (Série1)
- Personalizado:

Previsão

Avançar:  períodos

Recuar:  períodos

Definir Interseção =

Exibir Equação no gráfico

Exibir valor de R-quadrado no gráfico

Fechar

	A	B
1	10	128
2	144	241
3	332	396
4	540	558
5	656	656
6	784	764
7	976	915
8	1200	1092
9	1362	1228
10	1514	1349
11	1845	1618
12	2188	1894
13	2385	2059
14	2711	2321
15	2927	2483
16	3263	2705
17	3647	2897
18	3983	3039
19	4091	3080

$y = 0,7492x + 185,92$   
 $R^2 = 0,9944$



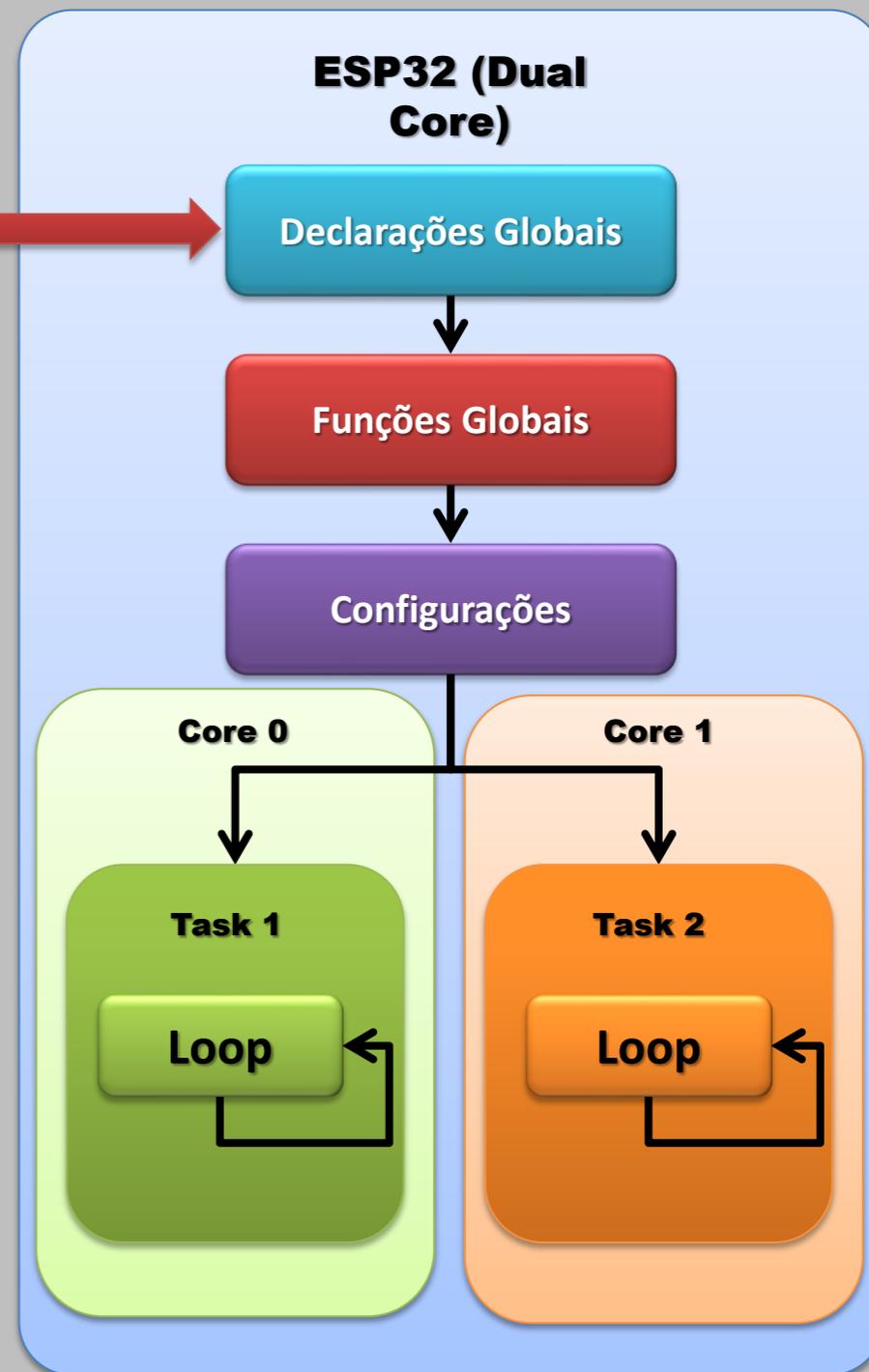
# Como calibrar a entrada de dados

O Excel gera a **equação linear**, que será usada no **código**.



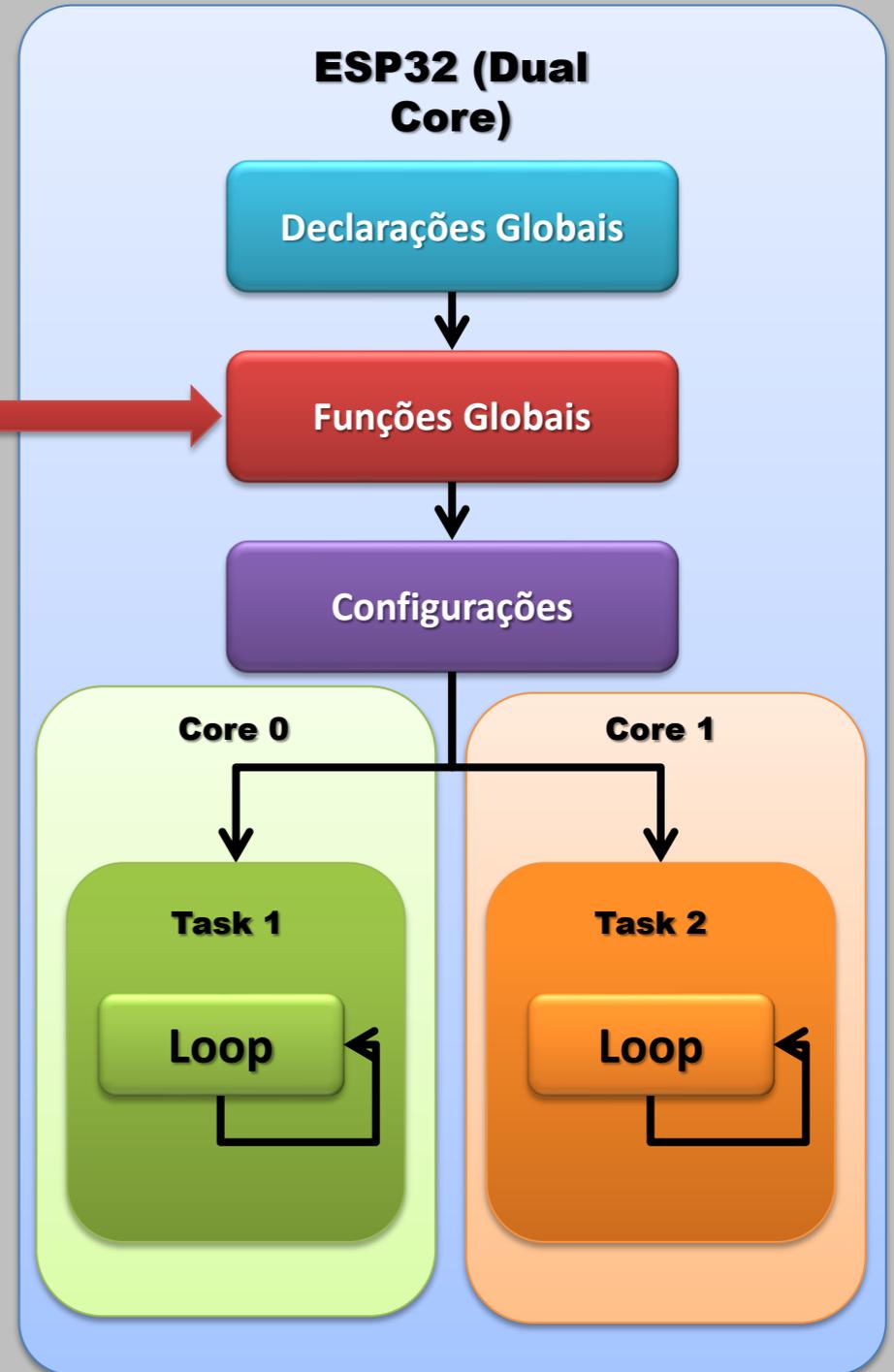
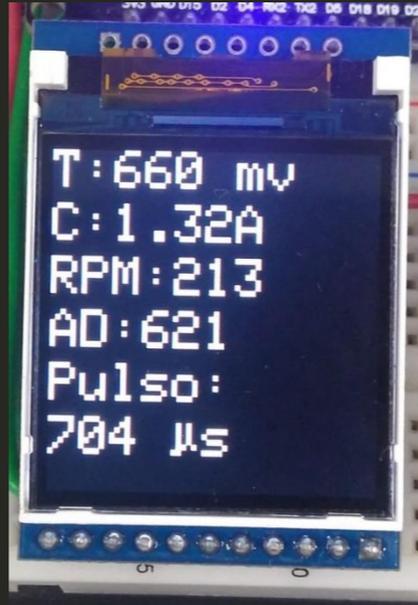
# Código...

```
1 #include <Adafruit_GFX.h> // Biblioteca gráfica principal
2 #include <XTronica1_ST7735.h> // Biblioteca específica do display
3 #include <SPI.h> // Biblioteca de comunicação SPI
4
5 // Configura os pinos que serão usados para a comunicação com o display
6 #define TFT_DC 2 // Seleção do registro
7 #define TFT_RST 4 // Pino de redefinição da tela
8 #define TFT_CS 5 // ativação do Display, se não ativado, não comunicará no barramento SPI
9
10
11 // Inicia a rotina de conversa com o display com os pinos configurados
12 // Como não colocamos o TFT_SCLK e o TFT_MOSI, a rotina presume que estamos usando o barramento SPI
13 Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);
14
15 //Instancia as Tasks
16 TaskHandle_t Task1, Task2;
17
18 //Constantes e variáveis -----
19
20 const int RST = 25; // Porta digital D25 - Reset do DRV8825
21 const int SLP = 26; // Porta digital D26 - Dormir (Sleep) DRV8825
22 const int ENA = 34; // Porta digital D34 - Ativa (enable) DRV8825
23 const int M0 = 35; // Porta digital D35 - M0 do DRV8825
24 const int M1 = 32; // Porta digital D32 - M1 do DRV8825
25 const int M2 = 33; // Porta digital D33 - M2 do DRV8825
26 const int DIR = 14; // Porta digital D14 - Direção (direction) do DRV8825
27 const int STP = 27; // Porta digital D27 - Passo (Step) do DRV8825
28 const int POTD = 13; // AD do potenciômetro do driver
29 const int POTV = 12; // AD do potenciômetro de velocidade do motor
30
31 int MeioPeriodo; // Meio período do pulso em microsegundos
32 float MMP; // Multiplicador de micropasso
33 int PPR = 200; // Número de passos por volta
34 int Pulsos; // Pulsos para o driver do motor
35 int Voltas; // voltas do motor
36 float RPM; // Rotacao por minuto
37 int leituraAd; // Valor da leitura do potenciômetro do driver
38 float tensaoPot; // Valor da tensão na saída do potenciômetro do driver
39 float correnteMot; // Valor da corrente configurada no driver
```



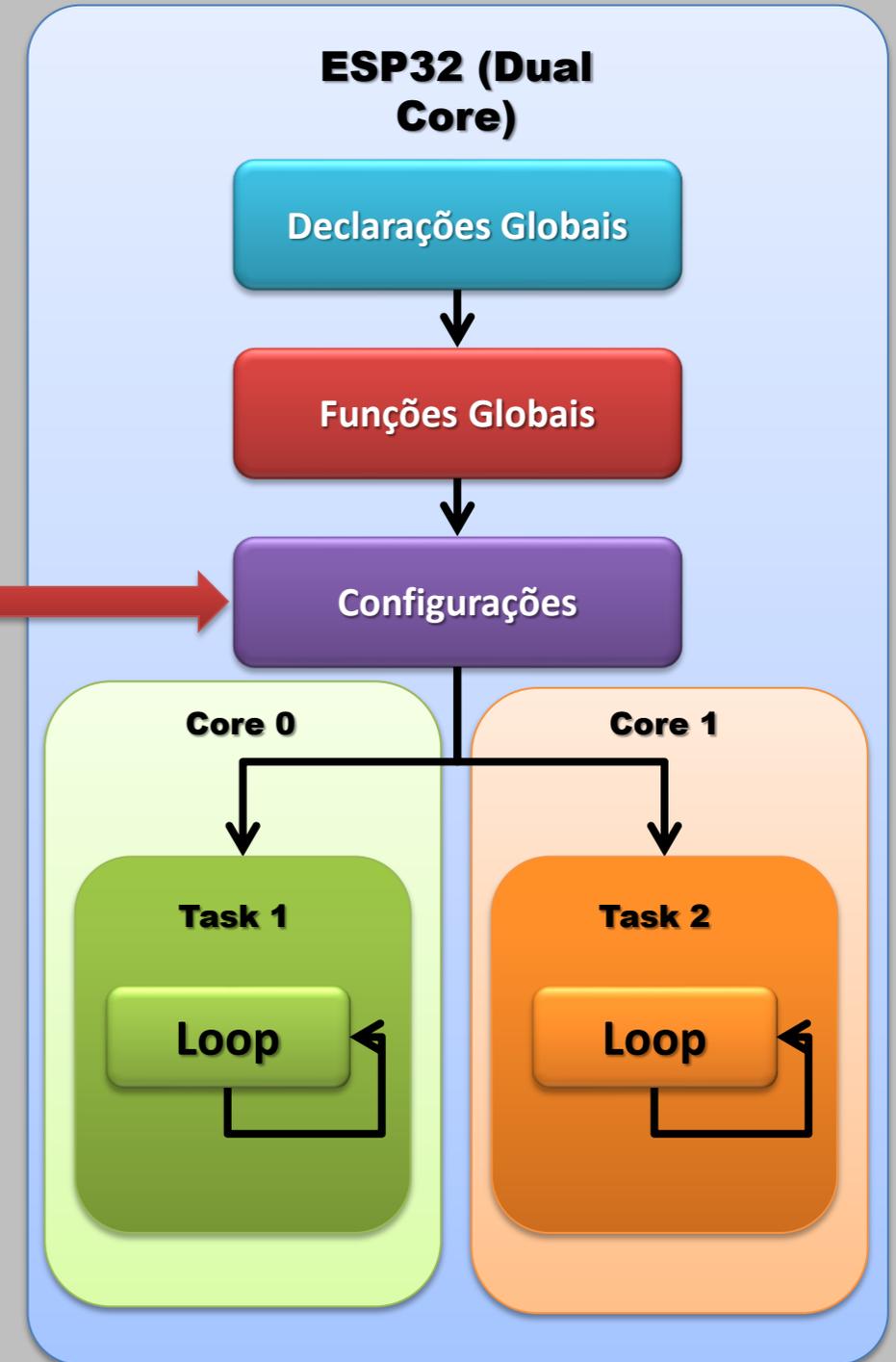
# Código...

```
41 // Funções -----
42
43 // Imprime os valores das variáveis no Display
44 void Imprime(){
45
46     // Cria um retângulo preto em cima dos dados impressos para apagá-los
47     tft.fillRect(24, 0, 48, 16, ST7735_BLACK);
48     // Posiciona o cursor na posição
49     tft.setCursor(24,0);
50     // Imprime os dados na posição do cursor
51     tft.print(int(tensaoPot));
52
53     tft.fillRect(24, 20, 48, 16, ST7735_BLACK);
54     tft.setCursor(24,20);
55     tft.print(correnteMot);
56
57     tft.fillRect(48, 40, 80, 16, ST7735_BLACK);
58     tft.setCursor(48,40);
59     tft.print(int(RPM));
60
61     tft.fillRect(36, 60, 48, 16, ST7735_BLACK);
62     tft.setCursor(36,60);
63     tft.print(analogRead(POTD));
64
65     tft.fillRect(0, 100, 48, 16, ST7735_BLACK);
66     tft.setCursor(0,100);
67     tft.print(MeioPeriodo);
68
69 }
70
71 // Calcula a corrente máxima suportada pelo driver
72 //Os dados dessa equação foram tiradas do datasheet do DRV8825
73 float calculoCorrente(float tensao) {
74     return (tensao / 1000) / (5 * 0.1); // 0.1: valor do resistor do driver
75 }
76
77 //Calcula o RPM do motor
78 float calculoRPM() {
79     //TPV: tempo por volta(segundos), PPR: passo por revolucao, MMP: multiplicador de micro passo
80     float TPV = (PPR * MMP * (MeioPeriodo * 2)) / 1000000;
81     //converte o TPV em RPM
82     return ((1.0 / TPV) * 60.0);
83 }
```



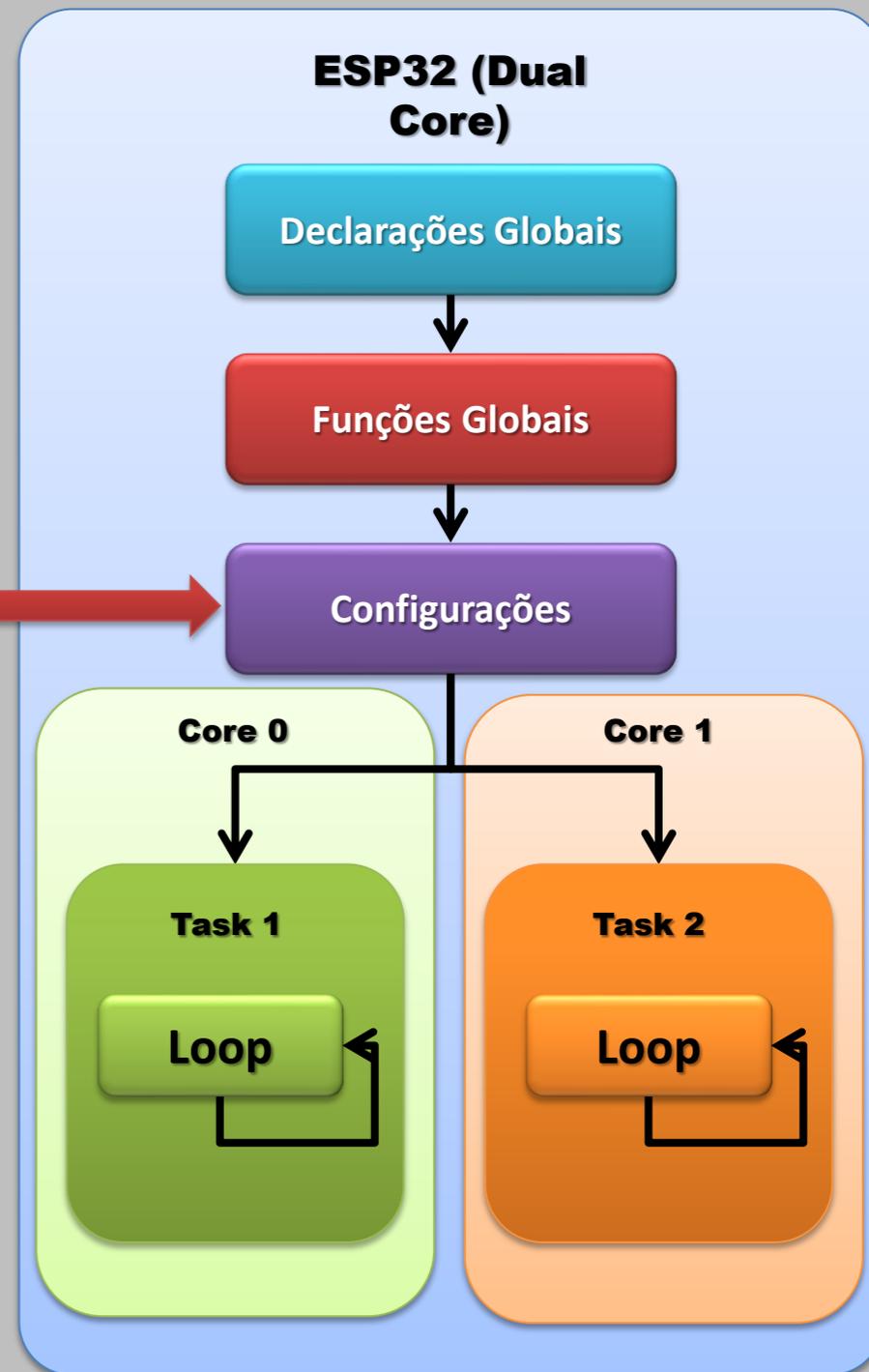
# Código...

```
79 void setup() {
80
81 //Inicia a comunicação serial
82 Serial.begin(115200);
83
84 // Configura as portas como saída
85 pinMode(RST, OUTPUT);
86 pinMode(SLP, OUTPUT);
87 pinMode(ENA, OUTPUT);
88 pinMode(M0, OUTPUT);
89 pinMode(M1, OUTPUT);
90 pinMode(M2, OUTPUT);
91 pinMode(DIR, OUTPUT);
92 pinMode(STP, OUTPUT);
93 // Configura as portas como entrada
94 pinMode(POTD, INPUT);
95 pinMode(POTV, INPUT);
96
97 //Configuração do driver-----
98
99 // MMP: Multiplicador de micro passo:
100 // 1- passo inteiro
101 // 2- meio passo
102 // 4- 1/4 de passo
103 // 8- 1/8 de passo
104 // 16 - 1/16 de passo
105 // 32 - 1/32 de passo
106 MMP = 1;
107
108 // Configura as portas do driver para passo inteiro
109 digitalWrite(M0, LOW);
110 digitalWrite(M1, LOW);
111 digitalWrite(M2, LOW);
112 // Desativa o modo sleep
113 digitalWrite(SLP, HIGH);
114 // Desativa o modo reset
115 digitalWrite(RST, HIGH);
116 // Ativa as saídas do driver
117 digitalWrite(ENA, LOW);
118 // Define a direção do motor
119 digitalWrite(DIR, HIGH);
```



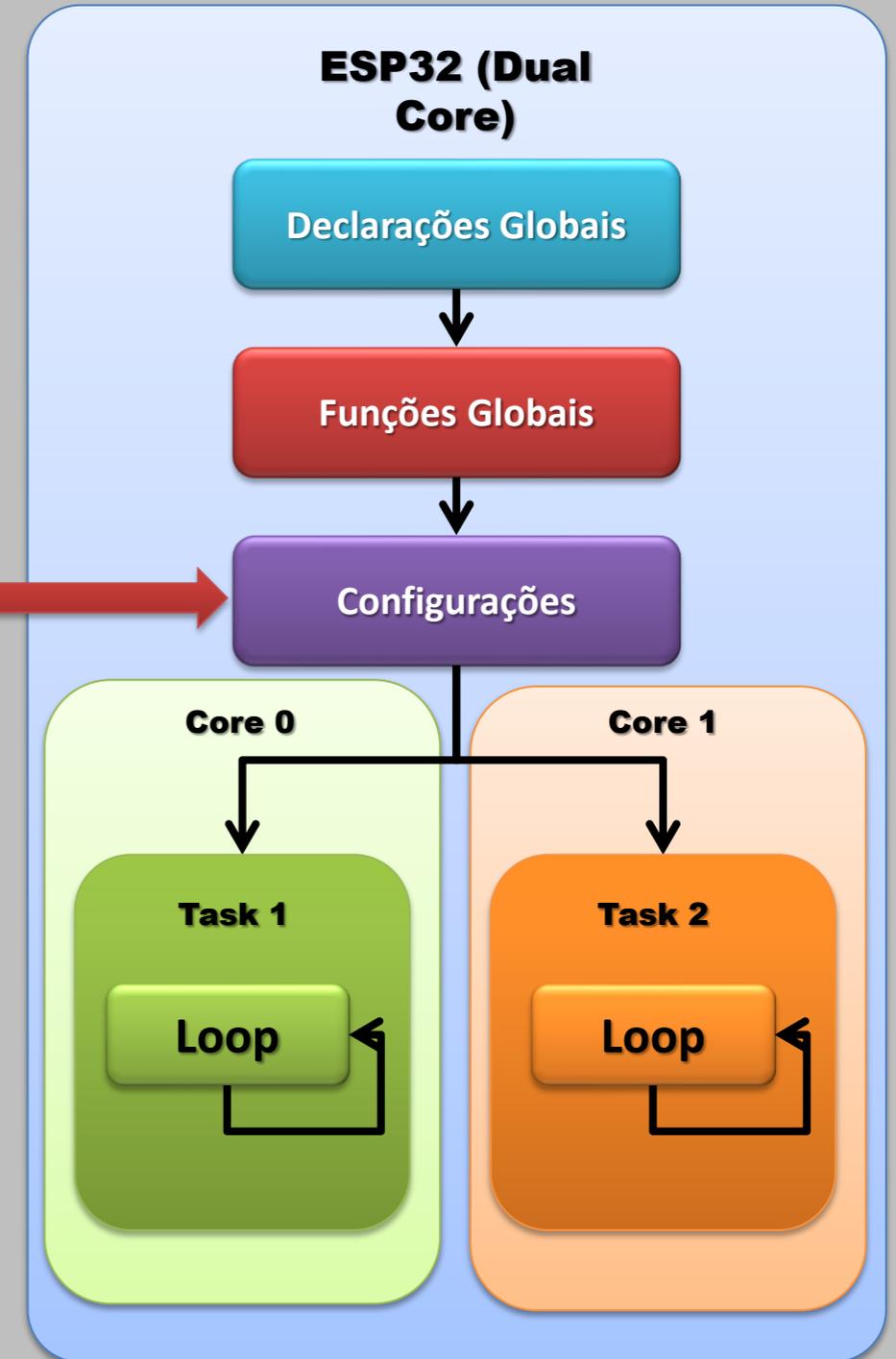
# Código...

```
127 //Configuração do Display -----
128
129 // Inicializa o Display
130 tft.init();
131
132 delay(100);
133
134 // Gira a tela em 180 graus
135 tft.setRotation(2);
136 // Define o tamanho do texto
137 tft.setTextSize(2);
138 // Define a cor do texto
139 tft.setTextColor(ST7735_WHITE);
140 // Define a quebra de linha do texto
141 tft.setTextWrap(false);
142 // Pinta a tela de preto
143 tft.fillScreen(ST7735_BLACK);
144
145 delay(500);
146
147 // Imprime as informações estáticas na tela
148 tft.setCursor(0,0);
149 tft.print("T:");
150 tft.setCursor(72,0);
151 tft.print("mv");
152
153 tft.setCursor(0,20);
154 tft.print("C:");
155 tft.setCursor(72,20);
156 tft.print("A");
157
158 tft.setCursor(0,40);
159 tft.print("RPM:");
160
161 tft.setCursor(0,60);
162 tft.print("AD:");
163
164 tft.setCursor(48,100);
165 // 229 é o código da letra "µ" na tabela ASCII
166 tft.print((char)229);
167 tft.print("s");
168
169 delay(500);
```



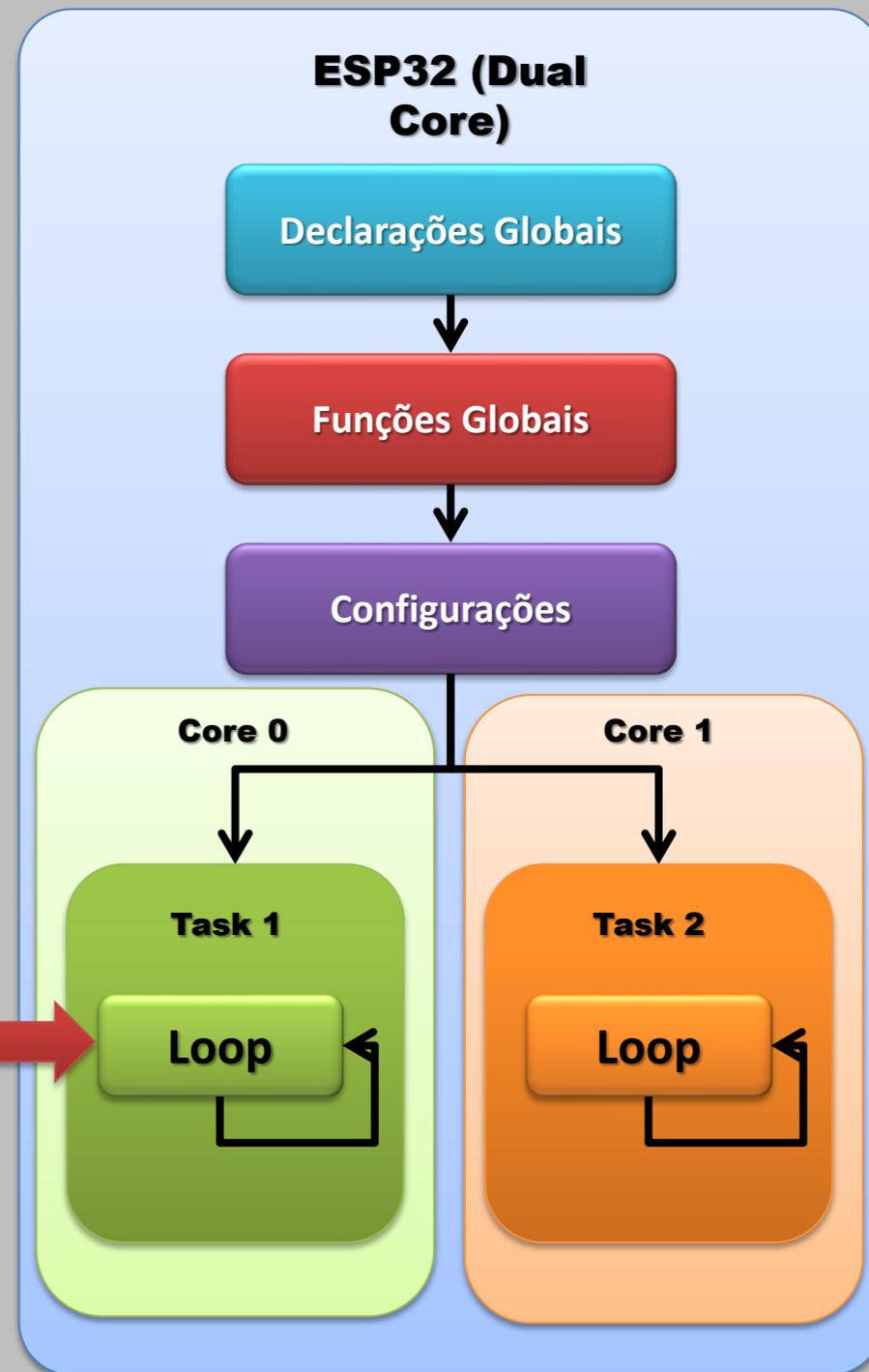
# Código...

```
171 // Configuração das tasks
172 xTaskCreatePinnedToCore(
173     codeForTask1,      // Função Task
174     "led1Task",       // Nome da Task
175     2000,              // Tamanho da pilha da Task
176     NULL,             // Parâmetro da Task
177     1,                // Prioridade da Task
178     &Task1,           // Identificação da Task para acompanhar a Task criada
179     0);               // Core usado
180
181 //Delay necessário para iniciar a task 1
182 delay(500);
183
184 xTaskCreatePinnedToCore(
185     codeForTask2,      // Função Task
186     "led2Task",       // Nome da Task
187     2000,              // Tamanho da pilha da Task
188     NULL,             // Parâmetro da Task
189     1,                // Prioridade da Task
190     &Task2,           // Identificação da Task para acompanhar a Task criada
191     1);               // Core usado
192
193 delay(500);
194 } // Fim do setup
195
```



# Código...

```
188 // Task 1 -----
189 void codeForTask1( void * parameter )
190 {
191     for (;;) {
192
193         // Lê o valor do potenciômetro de velocidade
194         MeioPeriodo = (analogRead(POTV)) / 2;
195         // Lê o valor do potenciômetro do driver
196         leituraAd = analogRead(POTD);
197         // Calcula a tensão da saída do potenciômetro do driver
198         tensaoPot = 0.7492*leituraAd + 185.92;
199         //Calcula a corrente máxima suportada pelo driver
200         correnteMot = calculoCorrente(tensaoPot);
201         //Calcula o RPM do motor
202         RPM = calculoRPM();
203         //Imprime as informações no display
204         Imprime();
205
206         delay(100);
207
208     }
209 }
```

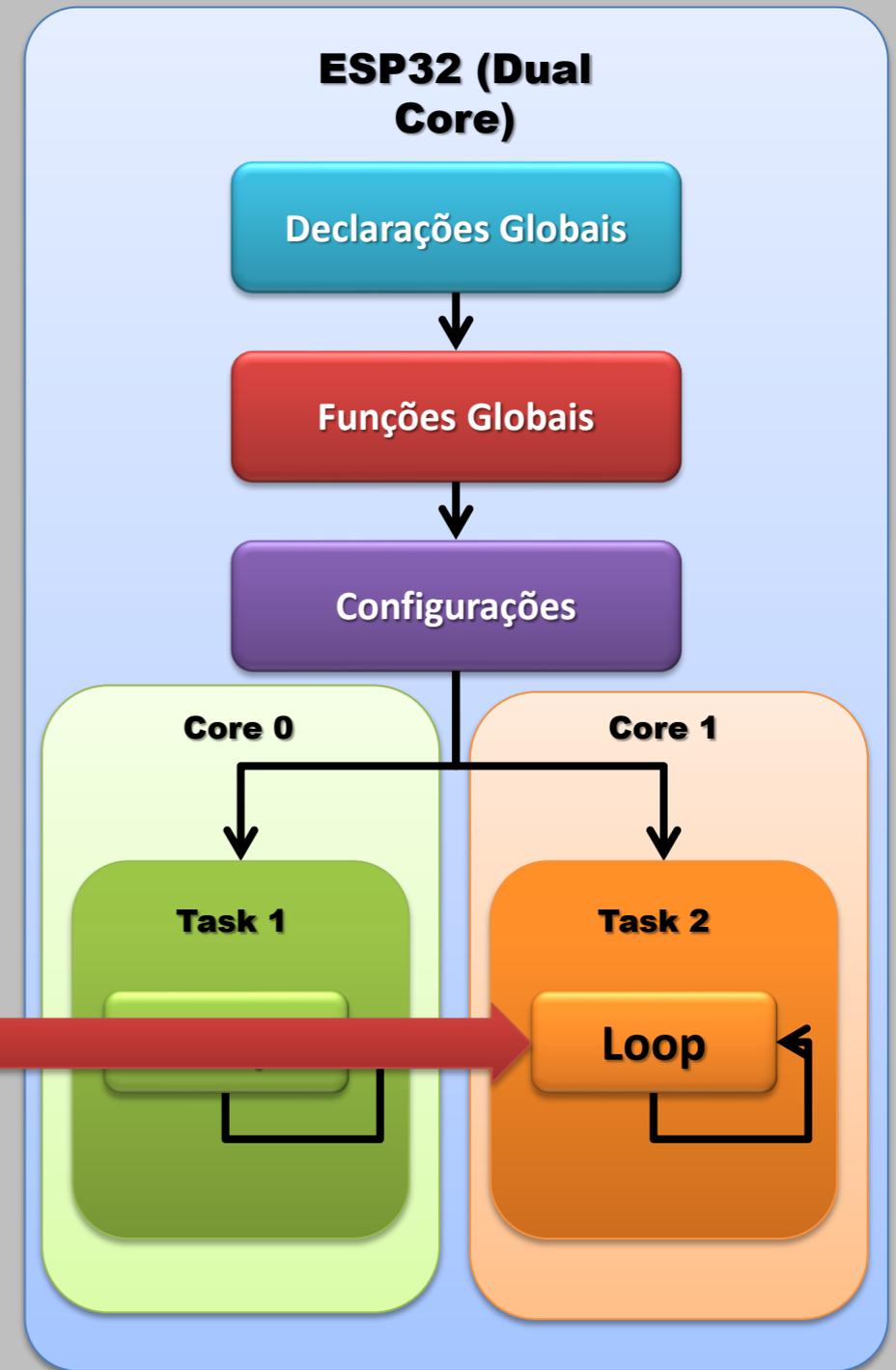


# Código...

```
210 // Task 2 -----  
211 void codeForTask2( void * parameter )  
212 {  
213     for (;;) {  
214         //Aqui é gerado o pulso do motor  
215         digitalWrite(STP, LOW);           // Pulso nível baixo  
216         delayMicroseconds(MeioPeriodo);   // MeioPeriodo de X microsegundos  
217         digitalWrite(STP, HIGH);          // Pulso nível alto  
218         delayMicroseconds (MeioPeriodo);  // MeioPeriodo de X microsegundos  
219     }  
220 }
```

## Observação:

```
222 // A função void loop() não será usada pelo programa  
223 // Porém é necessária para funcionar corretamente  
224 void loop() {  
225     delay(100);  
226 }  
227  
228 }
```



# Medindo a indutância dos motores de passo

	Nossas medidas do Nema 17	Datasheet do Nema 17
	Indutância	Indutância
Bobina 1	2,90mH	3mH
Bobina 2	6,38mH	



# Medindo a indutância dos motores de passo

Nossas medidas do Nema 17	
	Indutância
Bobina 1	2,90mH
Bobina 2	6,38mH



Datasheet do Nema 17	
Indutância	3mH

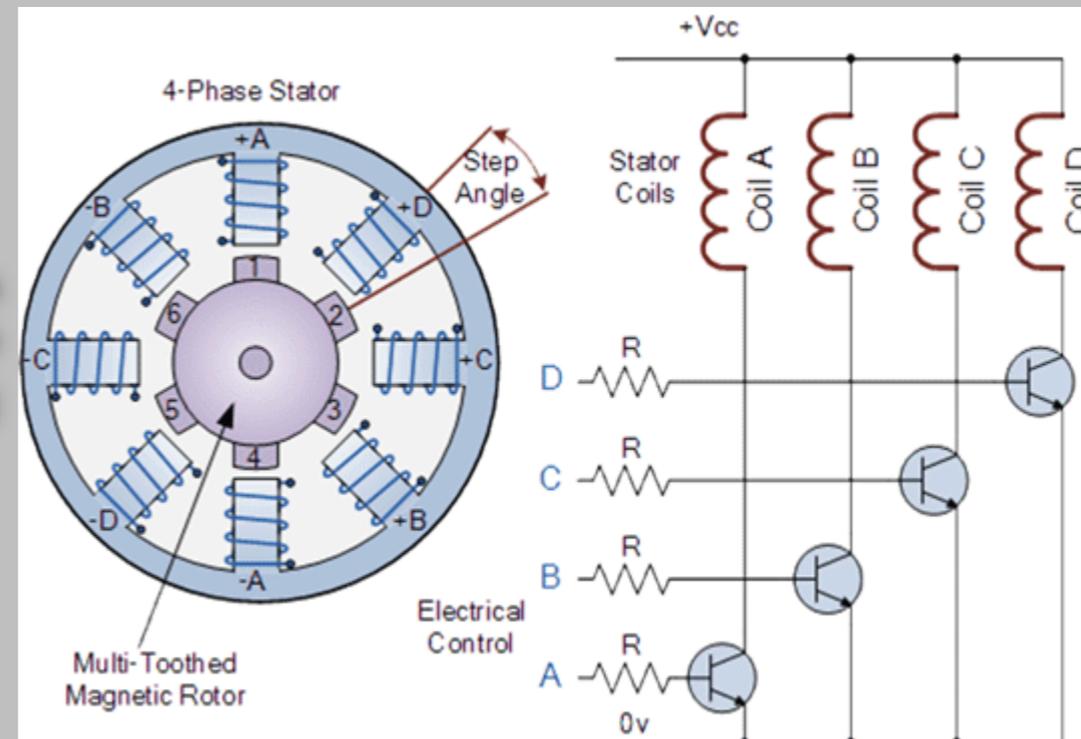
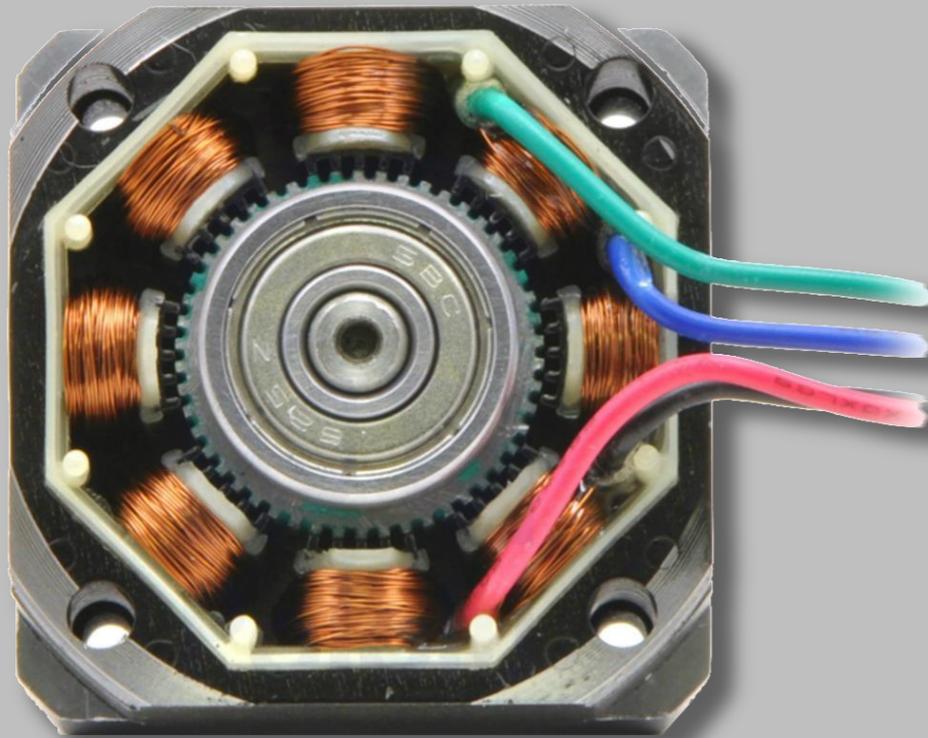
**Porque ocorre diferença  
entra as bobinas**



# Medindo a indutância dos motores de passo

## Porque ocorre diferença entre as bobinas?

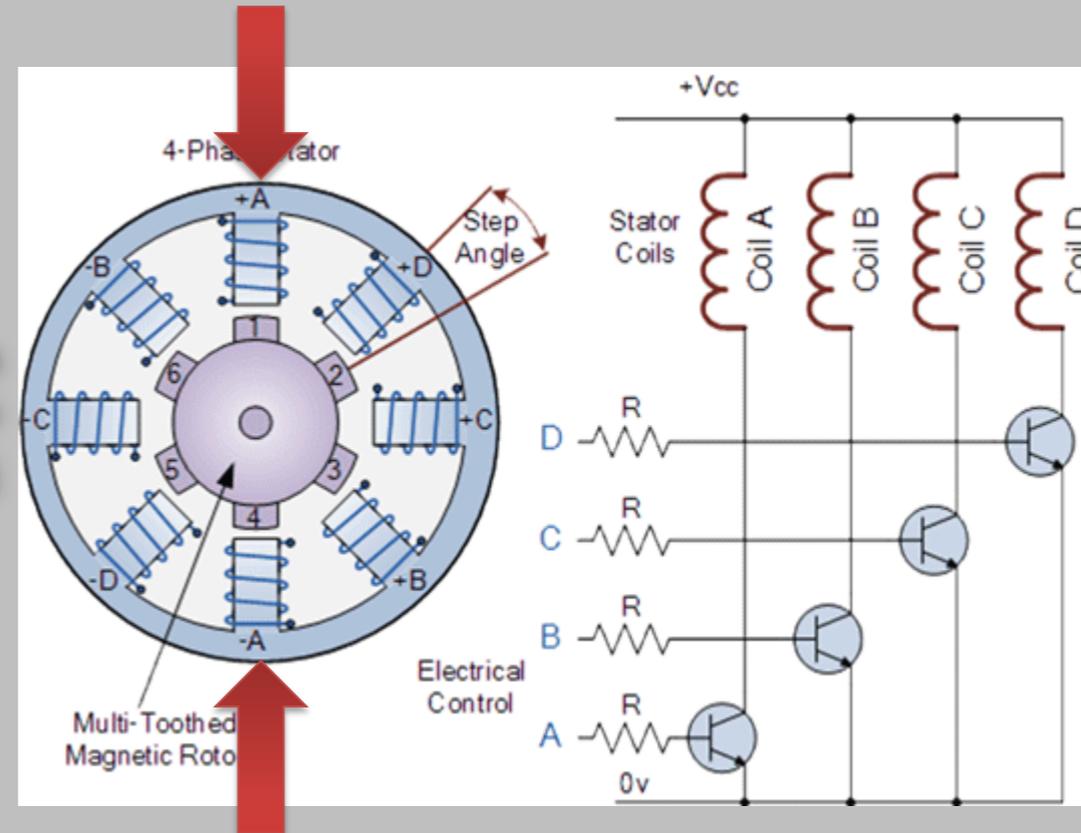
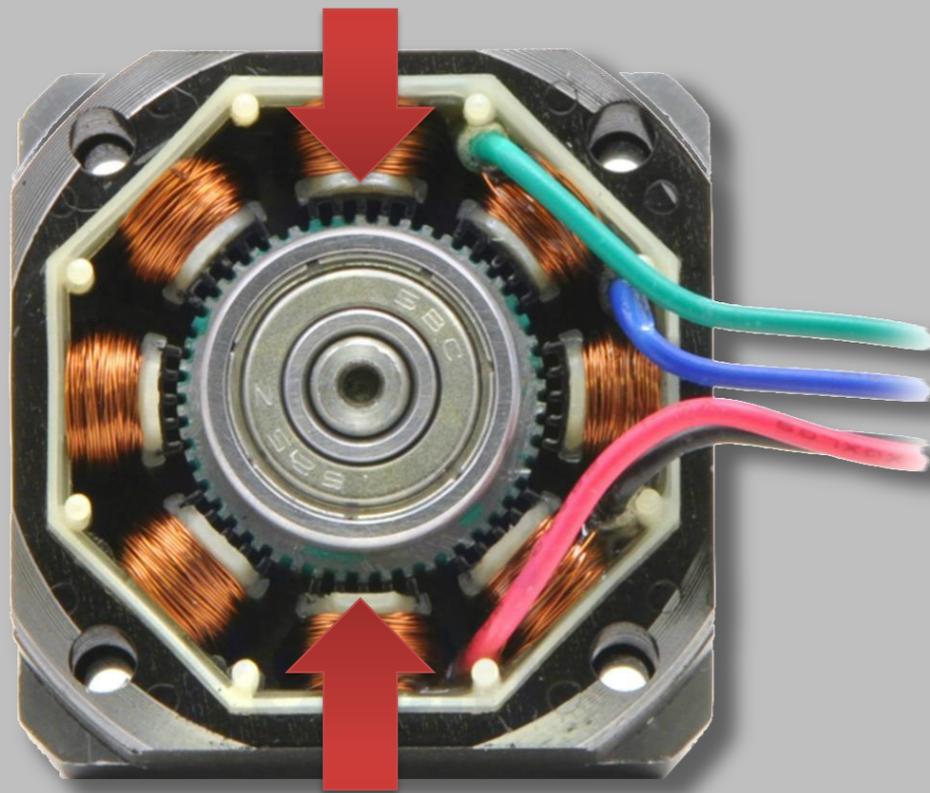
Devido ao **alinhamento** dessas bobinas com o **imã** do eixo, as medidas podem sofrer **alterações** pois o imã pode **incidir** no **campo magnético das bobinas**.



# Medindo a indutância dos motores de passo

## Porque ocorre diferença entre as bobinas?

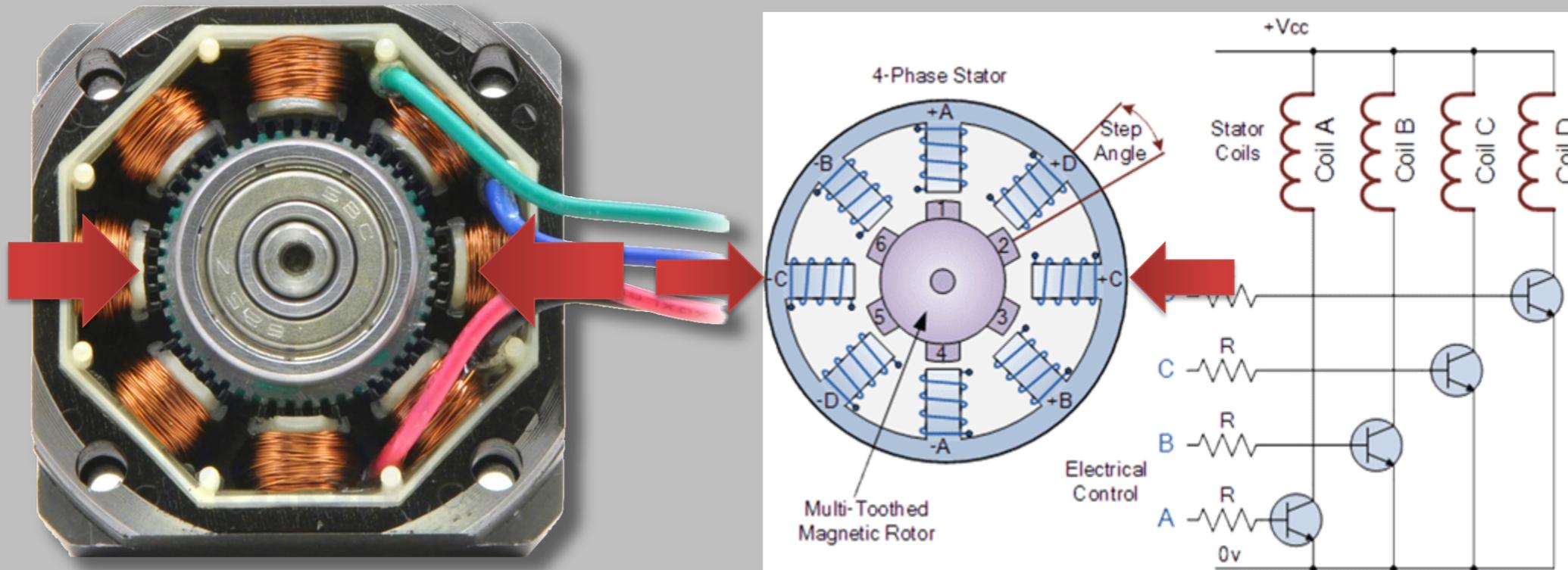
Repare que a bobina A está **alinhada** ao um conjunto de **imãs do eixo**, portanto o valor medido **maior** que das outras bobinas medidas



# Medindo a indutância dos motores de passo

## Porque ocorre diferença entre as bobinas?

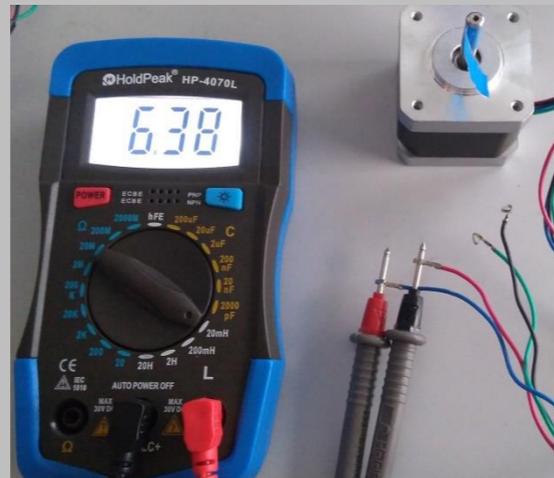
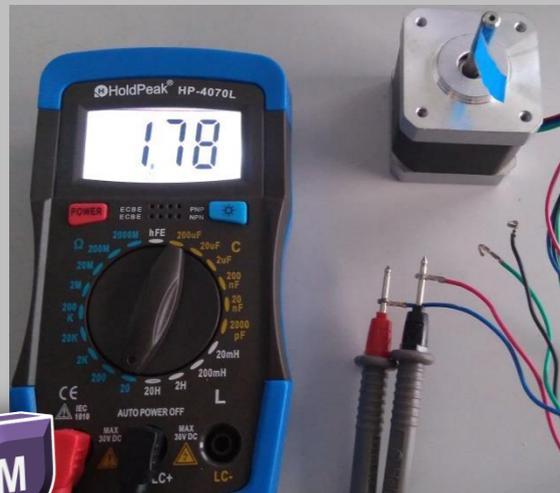
Quando as bobinas **não estão alinhadas** a um conjunto de **ímãs** do eixo, o valor medido é o valor **real** da indutância do motor, portanto o **menor valor medido**.



# Medindo Motores de passo

Nossas medidas do Nema 17		
	Resistência	Indutância
Bobina 1	1,94Ω	2,90mH
Bobina 2	1,78Ω	6,38mH

← **Menor medida**



**Datasheet do Nema 17**

Resistência

2Ω

Indutância

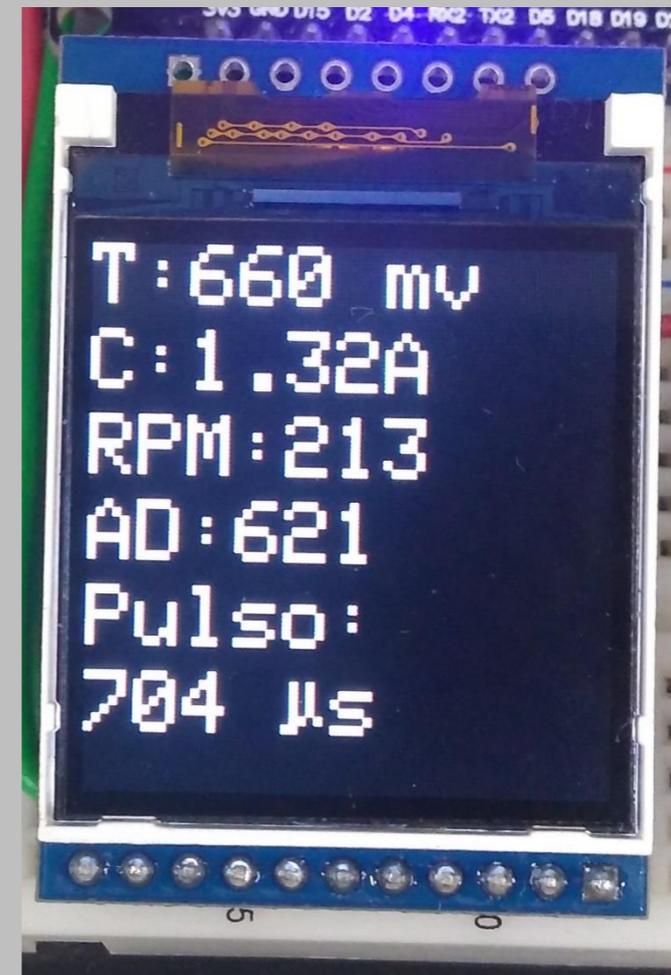
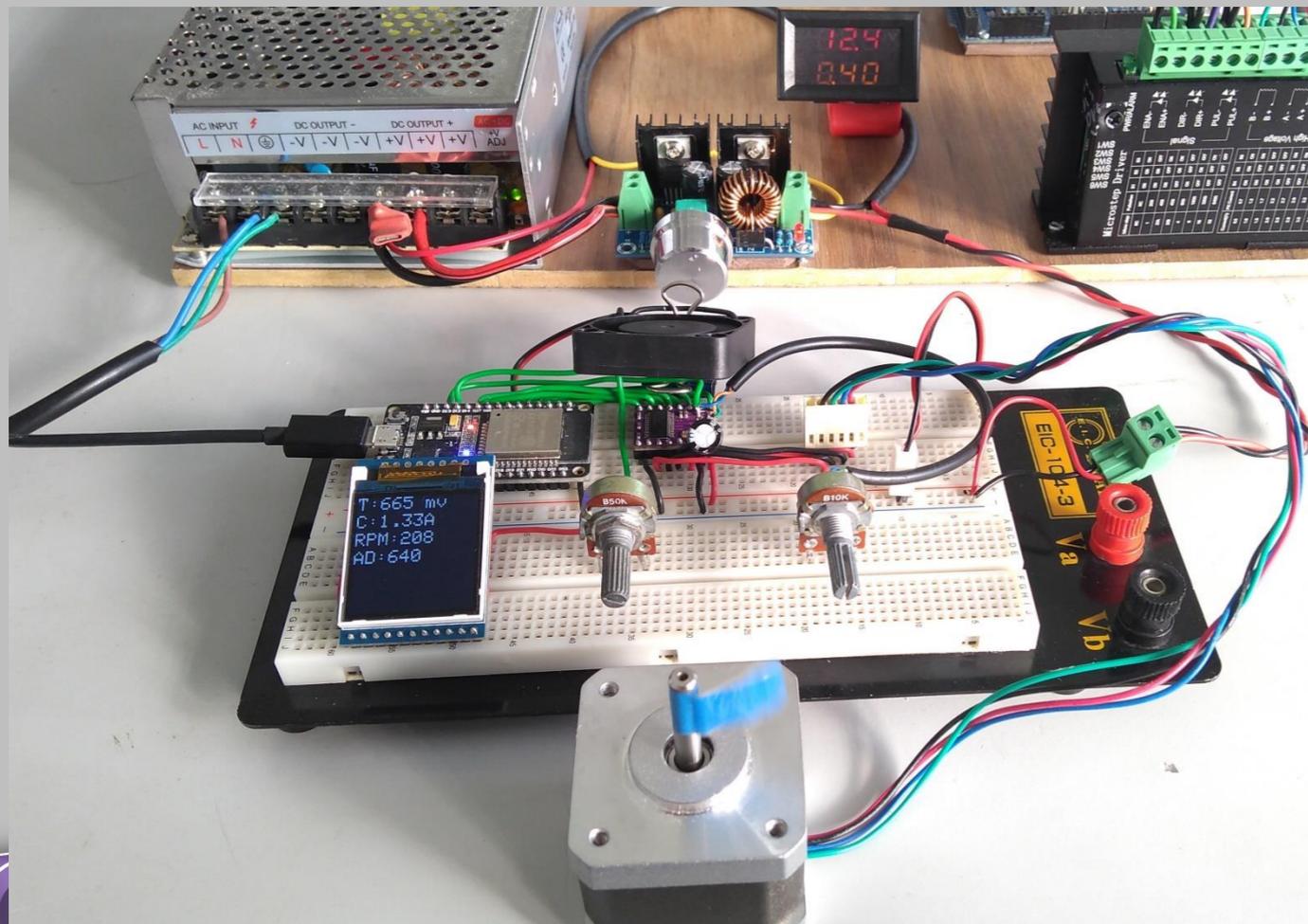
3mH



# Teste Prático

## Nema 17

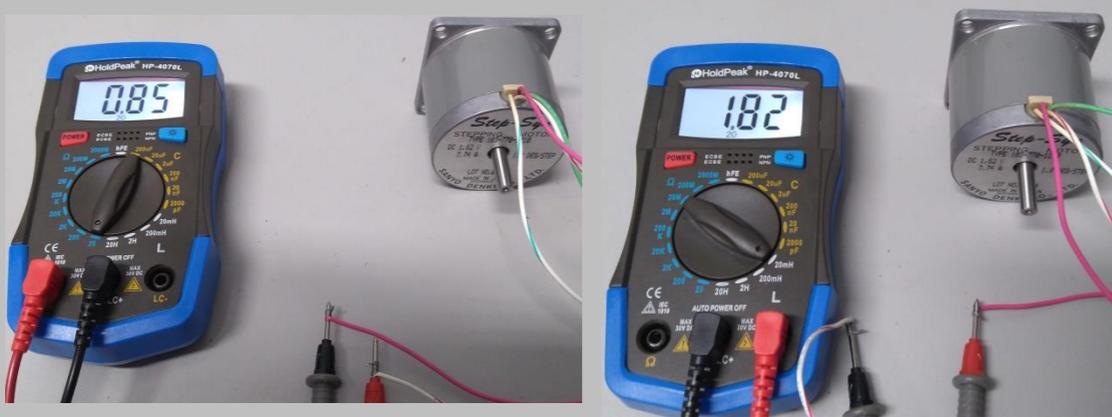
Tensão	Consumo	Tensão no Potenciômetro	Controle de Corrente do Driver	Giros por minuto (RPM)
12,4v	0,40A	660mv	1.32A	213



# Medindo Motores de passo

Nossas medidas do Motor grande de impressora		
	Resistência	Indutância
Bobina 1	0,85Ω	1,82mH
Bobina 2	0,86Ω	3,25mH

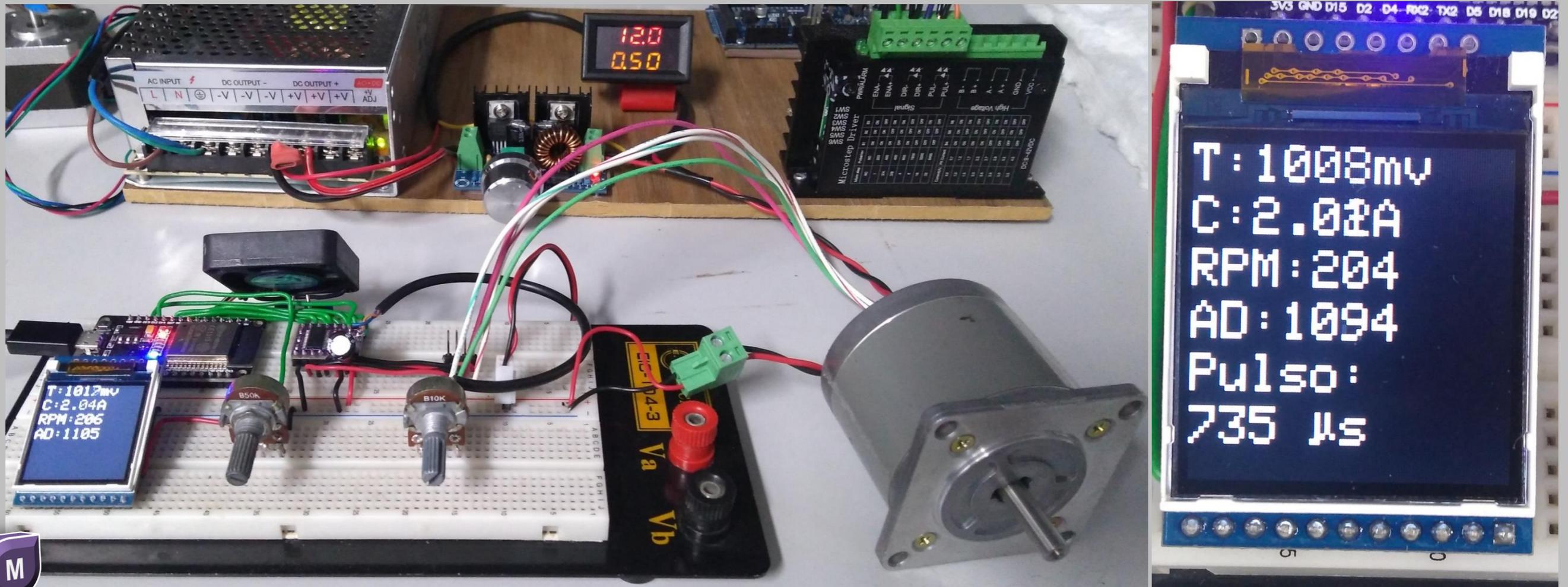
← Menor medida



# Teste Prático

## Motor Grande de Impressora

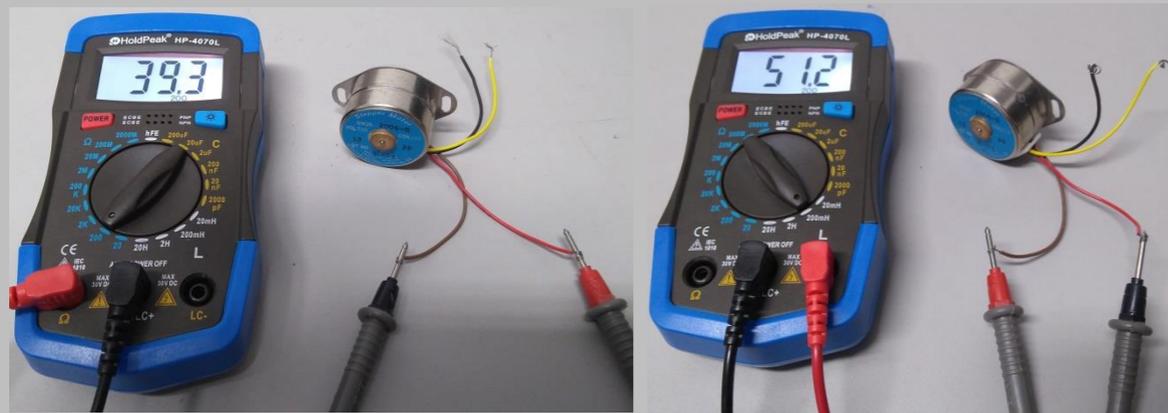
Tensão	Consumo	Tensão no Potenciômetro	Controle de Corrente do Driver	Giros por minuto (RPM)
12v	0,50A	1008mv	2,02A	204



# Medindo Motores de passo

Nossas medidas do Motor pequeno de impressora		
	Resistência	Indutância
Bobina 1	39,3Ω	51,2mH
Bobina 2	39,1Ω	79,0mH

Menor medida ←

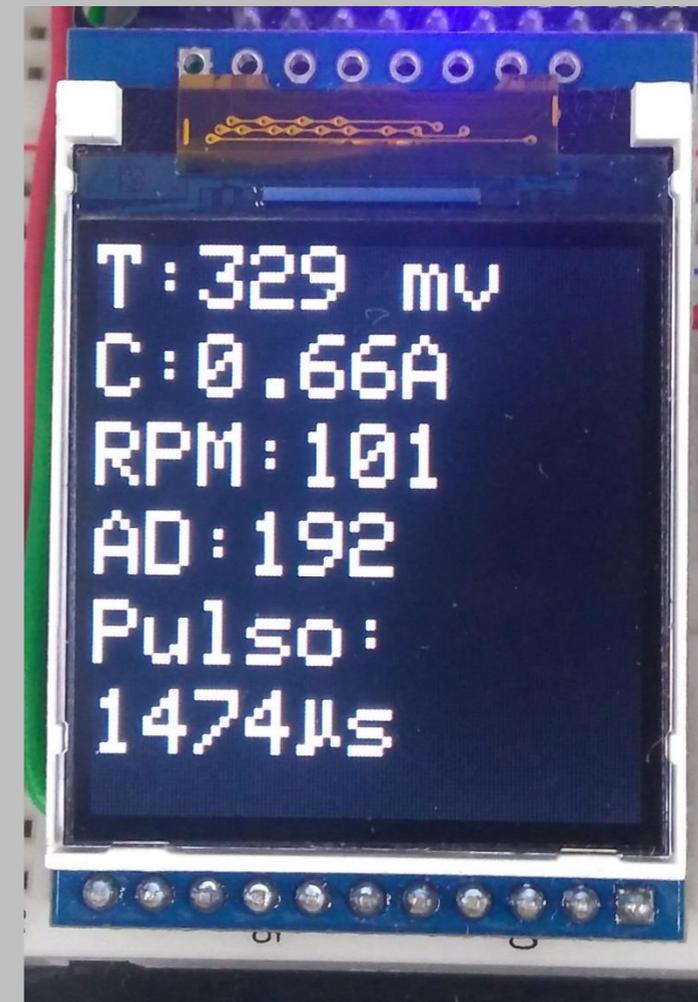
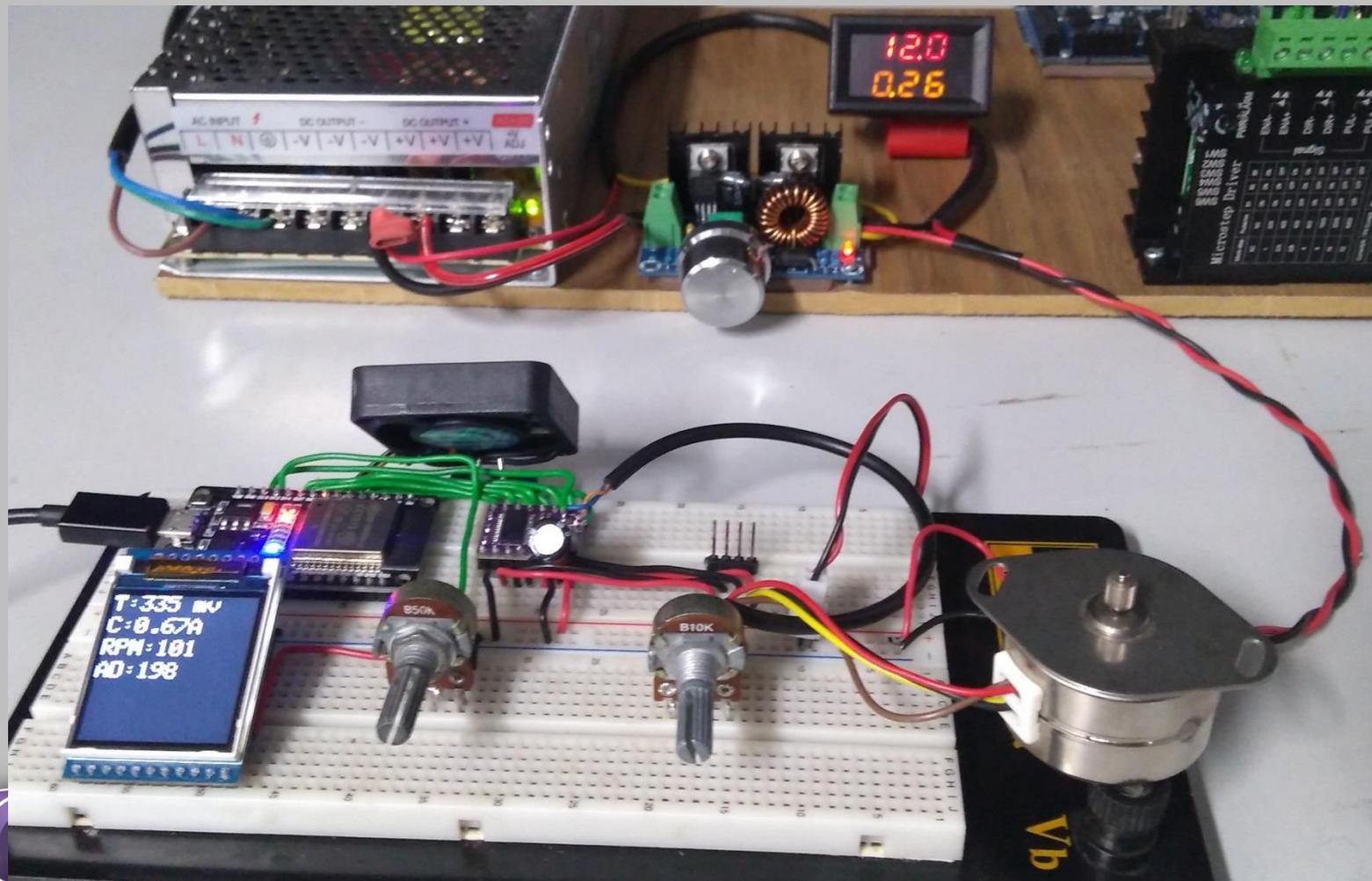


# Teste Prático

## Motor Pequeno de Impressora

Tensão	Consumo	Tensão no Potenciômetro	Controle de Corrente do Driver	Giros por minuto (RPM)
12v	0,26A	329mv	0,66A	1010*

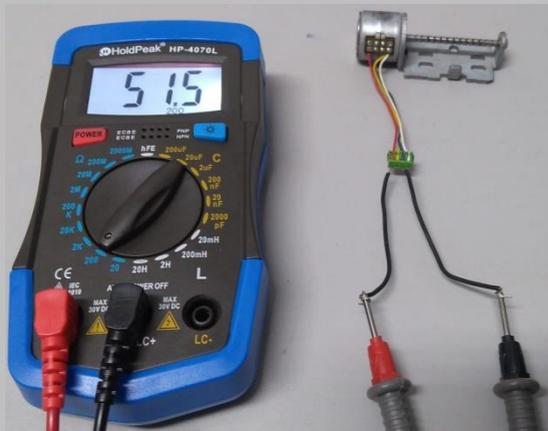
**\*Para motores de 20 passos por volta: RPM\*10**



# Medindo Motores de passo

Nossas medidas do Motor de Driver de DVD		
	Resistência	Indutância
Bobina 1	51,5Ω	29,7mH
Bobina 2	51,2Ω	26,9mH

← Menor medida

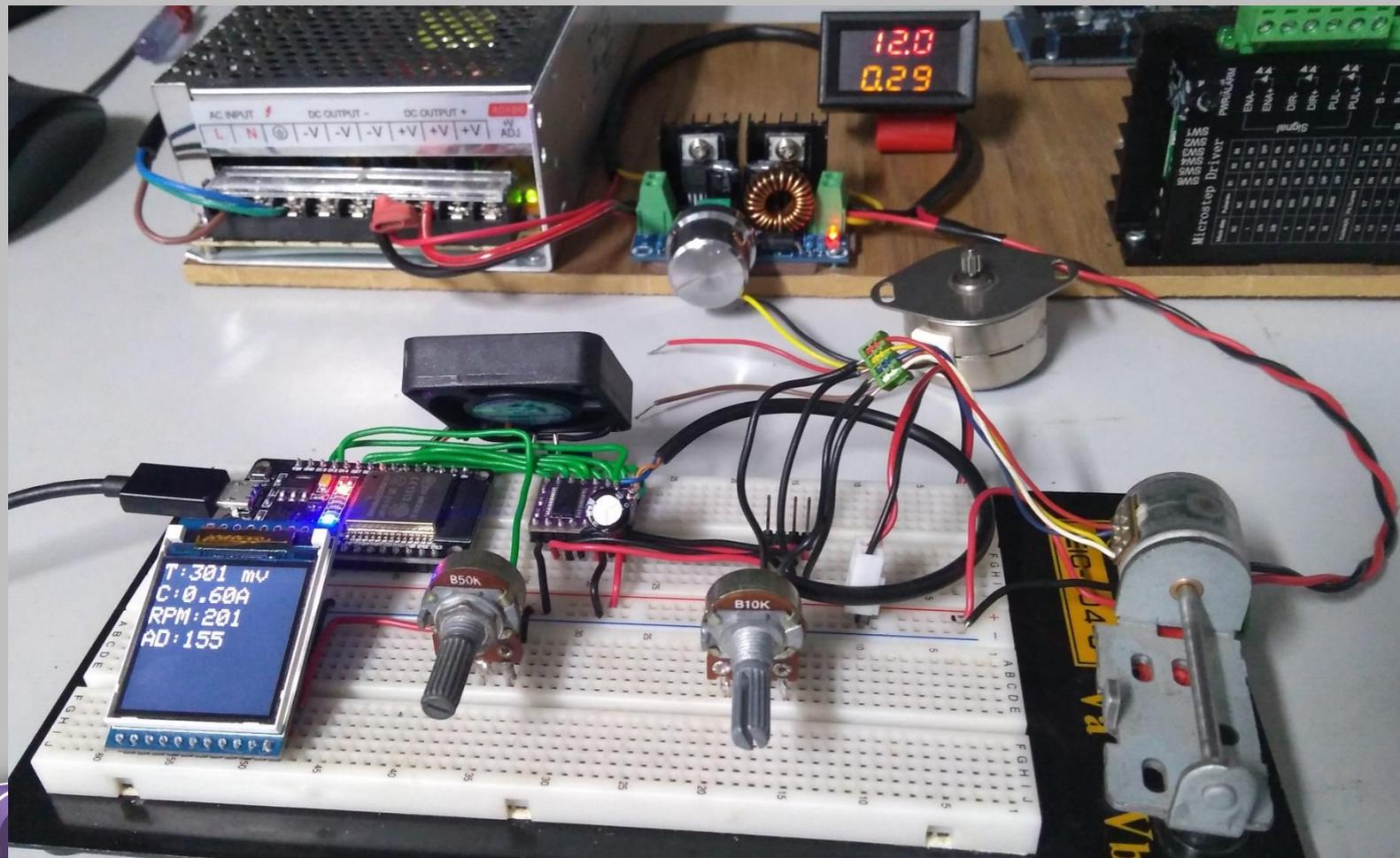


# Teste Prático

## Motor de driver de DVD

Tensão	Consumo	Tensão no Potenciômetro	Controle de Corrente do Driver	Giros por minuto (RPM)
12v	0,29A	322mv	0,64A	2060*

**\*Para motores de 20 passos por volta: RPM\*10**



Em [www.fernandok.com](http://www.fernandok.com)

Download arquivos PDF e **INO** do código fonte

