UNIVERSIDADE FEDERAL
DE UBERLÂNDIA
Faculdade de Engenharia Mecânica
Curso de Graduação em Engenharia Mecatrônica

Tópicos Especiais em Engenharia Mecatrônica II

Internet das Coisas Industriais – Profº José Jean-Paul Zanlucchi De Souza Tavares

Módulo 2 – Sistema de Controle Remoto de Pêndulo Invertido via IIoT

João Victor Martins Xavier.....12021EMT006
Marco Tulio Vilela Fonseca.....12021EMT016
Pedro Cabrini Costa Barros.....12021EMT009
Thalles Jonathan Trigueiro de Almeida.....11711EMT020

Uberlândia
2025

ÍNDICE

1. INTRODUÇÃO

O MQTT (Message Queuing Telemetry Transport) é um protocolo de comunicação leve e eficiente, baseado no modelo publish-subscribe, amplamente utilizado em sistemas de Internet das Coisas (IoT). Projetado para operar em ambientes com recursos limitados e largura de banda reduzida, o MQTT permite a troca de mensagens entre dispositivos e servidores (brokers) de forma escalável e confiável. Sua arquitetura centralizada, em que os dispositivos publicam dados em tópicos específicos e se inscrevem para receber informações relevantes, o torna ideal para aplicações que demandam baixo consumo energético e comunicação assíncrona, como monitoramento remoto, automação residencial e gestão de sensores em tempo real.

Para viabilizar a conexão do aplicativo IoT à rede, foi utilizada a biblioteca EWifi.h. Essa biblioteca simplifica o processo de autenticação na rede Wi-Fi da faculdade. Por meio de funções pré-configuradas, o EWifi.h estabelece uma conexão estável, mesmo em cenários com alta rotatividade de dispositivos ou interferências, garantindo que o aplicativo permaneça vinculado ao broker MQTT sem interrupções críticas.

Uma vez conectado à rede Wi-Fi institucional, o dispositivo IoT estabelece comunicação com o broker MQTT, atuando como cliente para publicar dados de sensores ou subscrever a comandos de controle. A escolha do broker adequado — como o Mosquitto (utilizado no serviço da Google) —, combinada à configuração de tópicos hierárquicos e níveis de QoS (Quality of Service), assegura a entrega eficiente de mensagens, mesmo em redes instáveis. Essa integração entre EWifi.h e MQTT não apenas viabiliza a interoperabilidade entre dispositivos heterogêneos, mas também cria uma base robusta para soluções IoT escaláveis, alinhadas às demandas de ambientes acadêmicos e industriais.

2. OBJETIVOS

Neste módulo, aplicaremos os conceitos de Internet das Coisas Industriais (IIoT) para o desenvolvimento de um sistema de controle remoto de um pêndulo invertido. Iremos projetar e implementar um aplicativo capaz de se comunicar com sensores e atuadores, monitorando o estado do sistema em tempo real e enviando comandos de controle. O aplicativo será utilizado para viabilizar a realização de aulas remotas de laboratório da disciplina de Controle Linear da Universidade Federal de Uberlândia (UFU), permitindo que os estudantes interajam com o experimento de forma prática e segura, mesmo a distância. Este módulo visa proporcionar uma compreensão prática da aplicação de tecnologias IIoT em sistemas dinâmicos e críticos, além de ampliar o acesso às práticas laboratoriais.

3. MATERIAIS E MÉTODOS

3.1. Materiais utilizados

Para o desenvolvimento do projeto e implementação do sistema de controle remoto do pêndulo invertido, utilizou-se um microcontrolador ESP32, responsável pelo processamento e pela comunicação via Wi-Fi. Foram empregados fios de conexão para a integração dos componentes eletrônicos e uma fonte de alimentação DC para o fornecimento de energia ao circuito. Um motor DC foi utilizado para o acionamento do sistema mecânico, controlado por meio de uma ponte H modelo L298N. A montagem dos circuitos foi realizada sobre uma *protoboard*, permitindo fácil adaptação e organização dos elementos. Também foram utilizadas peças específicas para a construção do pêndulo invertido, compostas por rodas, haste e conexões estruturais.

No desenvolvimento do sistema de controle, foi utilizado um computador para a programação e testes do microcontrolador ESP32, empregando a IDE do Arduino configurada para o ambiente ESP32. Para a criação da interface de comunicação entre o usuário e o sistema de controle, foi desenvolvido um aplicativo utilizando a plataforma .NET MAUI, possibilitando o monitoramento e o envio de comandos ao pêndulo de forma remota e prática.

3.2. Implementação do aplicativo

A criação do aplicativo para controle remoto do pêndulo invertido foi realizada utilizando a plataforma .NET MAUI no Visual Studio 2022. O processo de desenvolvimento pode ser dividido nas seguintes etapas:

- **Configuração do ambiente**
  - Instalação do Visual Studio 2022.
  - Instalação dos pacotes necessários para o desenvolvimento com .NET MAUI.
  - Criação de um novo projeto do tipo MAUI App.
- **Desenvolvimento da interface gráfica**
  - Inserção de caixas de texto para a entrada dos parâmetros de controle (ganho, ângulo de referência, atraso, entre outros).
  - Adição de botões para estabelecer a conexão com o broker MQTT e enviar os parâmetros ao sistema de controle
  - Organização da interface para garantir usabilidade e praticidade para o usuário, deixando cada tópico do roteiro de laboratório dividido em páginas.

- **Implementação do backend**
  - Desenvolvimento da lógica de comunicação utilizando a extensão MQTTnet [3] para .NET, que facilita a implementação do protocolo MQTT no aplicativo.
  - Criação das rotinas de manipulação dos parâmetros inseridos pelo usuário.
  - Estabelecimento da comunicação com o broker MQTT, hospedado em uma máquina virtual (VM) no Google Cloud.
  - Implementação de tratamento de erros e verificação da conexão para garantir a robustez da comunicação.
  - Foram criadas duas páginas utilizando *xaml*, sendo elas ProportionalControlPage e LeadLagControlPage, ambas utilizam a mesma lógica na qual é requerido estar conectada com o broker para os botões dos parâmetros enviarem os parâmetros e posteriormente o ESP-32 subscrever a mensagem ao broker, e nelas será possível monitorar a angulação do pêndulo quando em funcionamento. Assim como mostrará as figuras *1* e *2*, abaixo.
  - Há também uma classe essencial para funcionamento, a MqttService.cs, na qual é feita toda a lógica de conexão com o broker para publicar e subscrever via Mqtt, utilizando a extensão MQTTNet e métodos gerados a partir dela. Essa página é de suma importância, pois é nela também que é possível fazer a alteração do broker MQTT que será utilizado para envio dos dados:

    [...
    private IMqttClient _mqttClient;
    private readonly string _server = "34.151.193.145";
    private readonly int _port = 1883;
    ...]

  - Todo o código fonte do aplicativo e os códigos dos controladores, podem ser encontrados no repositório: https://github.com/marcotuliovf/IIoT-Controle/
- **Testes e validação**
  - Realização de testes para validar a transmissão dos parâmetros entre o aplicativo, o broker e o ESP32.
  - Verificação da estabilidade da comunicação em diferentes condições de rede.
  - Avaliação do funcionamento remoto do sistema, assegurando a correta operação do pêndulo invertido, após envio correto dos parâmetros via aplicativo.

3.2.1. Como utilizar o aplicativo

Antes de começar, deve-se seguir o roteiro de laboratório fornecido pelo professor da disciplina de Controle Linear. Ao tentar implementar o controlador proporcional, o aluno irá perceber que não é possível utilizá-lo para este experimento. Portanto, deverá usar o controlador Lead, que será o foco do processo a seguir.

- Abrindo o aplicativo

  Ao abrir o aplicativo, você será apresentado com duas abas principais:

  - Controlador Proporcional (padrão aberto)
  - Controlador Lead

- Observação comum às duas abas
  - Ambas as abas possuem um compasso, que indica em tempo real o ângulo atual do pêndulo invertido. O valor do ângulo é atualizado constantemente conforme o movimento do pêndulo.
- Usando o Controlador Proporcional
  - Na aba do Controlador Proporcional, você encontrará duas caixas de entrada:
    - Ângulo de Referência (º)
    - Ganho Proporcional (Kp)
- Também há dois botões:
  - Conectar ao Broker: Clique neste botão primeiro para estabelecer a conexão com o Broker MQTT.
  - Enviar Parâmetros: Após a conexão bem-sucedida com o Broker, você pode inserir os parâmetros nas caixas de entrada e clicar em Enviar Parâmetros.

Quando a conexão for realizada com sucesso, uma mensagem de confirmação será exibida, indicando que você pode prosseguir com o envio dos parâmetros.

- Usando o Controlador Lead
  - Para acessar a aba do Controlador Lead,basta clicar na segunda aba.
  - Na aba Lead, você encontrará cinco caixas de entrada:
    - Ângulo de Referência (graus)
    - Ganho (k)
    - Parâmetro a
    - Parâmetro b

- Atraso (ms)

– Assim como na aba do controlador proporcional, há dois botões:

- Conectar ao Broker: Estabeleça a conexão com o Broker MQTT primeiro.
- Enviar Parâmetros: Após a conexão, insira os valores nas caixas e clique em Enviar Parâmetros.

Esse processo garante que você esteja enviando os parâmetros corretos para o controlador conforme as configurações inseridas no aplicativo.

Abaixo se encontra as imagens para melhor entendimento da interface do aplicativo:

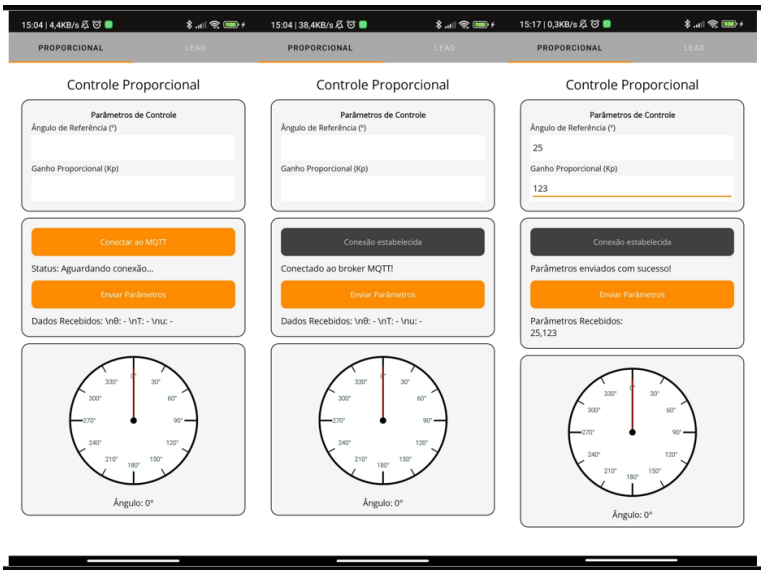

Figura 1 – Interface para conexão e envio de parâmetros do controlador Kp – Fonte: do próprio autor

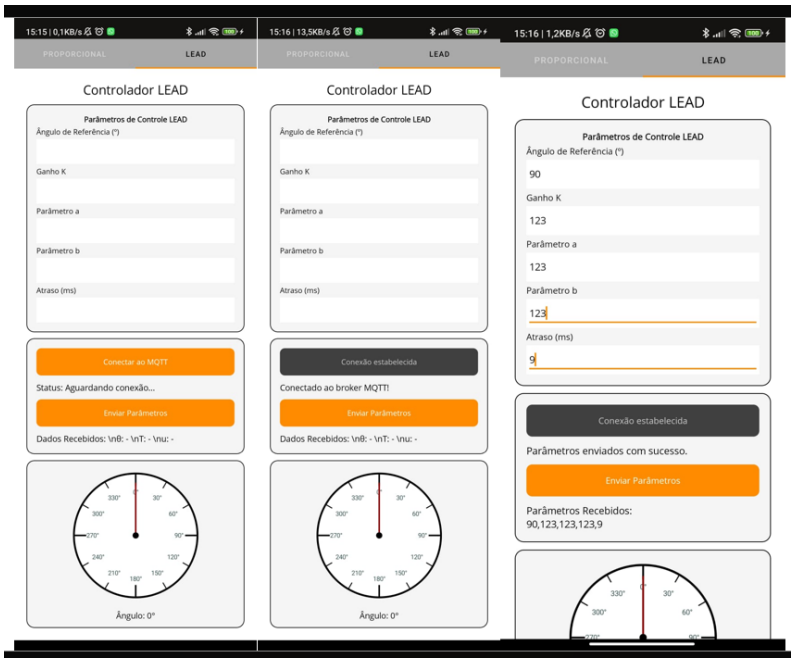

Figura 2 – Interface para conexão e envio de parâmetros do controlador LEAD – Fonte: do próprio autor

Como pode ser observado nas Figuras 1 e 2, a primeira exibe a aba correspondente à implementação do controlador proporcional para tentar controlar o pêndulo, enquanto a segunda apresenta a aba referente à segunda parte do roteiro, que trata da implementação do controlador LEAD. Elas demonstram a visão ao abrir o aplicativo, estabelecer conexão com o broker, e após o envio dos parâmetros, respectivamente. Posteriormente, com o sistema de controle ligado, será possível rastrear o ângulo que o pêndulo corresponderá em tempo real.

3.3. Implementação dos controladores

A implementação dos controladores ESP-32 se deu através de códigos feitos na IDE do Arduino, nos quais são os códigos do proporcional em anexo no Apêndice 1 e do LEAD em anexo no Apêndice 2. Além disso foi utilizado uma dependência WifiPassword.h para que fosse possível a conexão com o wi-fi da UFU via autenticação, facilitada por meio da biblioteca EWifi.h. Além disso a WifiPassword.h é crucial, pois nela também é possível alterar o broker utilizado para conexão:

```
/** Input WiFi data
 * Definition of macros SSID, username and password
*/
#define SSID1 "UFU-Institucional"
#define SSID2 "eduroam"
#define SSID3 ""
#define password ""
#define username "email-institucional"
#define userpassword "senha"
#define anonymous "anonymous@ufu.br"

/** Input WiFi data
 * Definition of macros SSID, username and password
*/
#define MQTTServer "34.151.193.145"
#define MQTTPort 1883

/* DECLARATION OF TOPICS VARIABLES */
const char* topic = "pendulo/parametroskp";
```

4. RESULTADOS ENCONTRADOS

O desenvolvimento do aplicativo para controle remoto do laboratório de controle linear do pêndulo invertido na UFU apresentou desafios técnicos significativos, mas culminou em resultados satisfatórios, cumprindo o objetivo principal de viabilizar a operação remota do sistema. A seguir, destacam-se os principais pontos observados durante a execução do projeto:

4.1. Desafios Técnicos Encontrados

Conexão MQTT com o Broker:

Inicialmente, foram identificadas dificuldades na comunicação entre o aplicativo e o broker MQTT, essencial para transmissão de dados em tempo real. Problemas de latência, autenticação e instabilidade na conexão comprometiam a sincronização entre o usuário remoto e o equipamento físico. A solução envolveu a revisão do código de conexão, a implementação de tratamento robusto de exceções e a otimização dos parâmetros de rede, resultando em uma comunicação estável e confiável, a partir da utilização da extensão MQTTNet.

Problemas com ESP32 e Motores:

Durante os testes, observou-se mau funcionamento intermitente dos microcontroladores ESP32, além de inconsistências no acionamento dos motores responsáveis pelo movimento do pêndulo. Suspeitou-se de falhas na alimentação elétrica e desgaste físico dos componentes. Para contornar o problema, substituíram-se os módulos ESP32 com defeito por unidades revisadas e os motores originais por modelos compatíveis de reposição, garantindo maior precisão na resposta do sistema. Adicionalmente, realizou-se a recalibração dos drivers de motor e a atualização do firmware dos dispositivos, assegurando compatibilidade com o aplicativo.

4.2. Funcionamento do Sistema Remoto

Após a resolução dos entraves técnicos, o sistema foi integrado e testado em condições operacionais. O aplicativo demonstrou capacidade de estabelecer conexão segura e estável com o laboratório físico via internet e capacidade de comunicação, troca de dados e comandos de controle em tempo real.

4.3. Avaliação Final

Os resultados dos testes realizado em laboratório comprovaram a viabilidade do controle remoto do pêndulo invertido, com desempenho comparável às operações presenciais. A

substituição de componentes físicos e a robustez da comunicação MQTT foram determinantes para o sucesso, garantindo replicabilidade do modelo em outros laboratórios da instituição.

5. CONCLUSÃO

O desenvolvimento do aplicativo para controle remoto do pêndulo invertido superou desafios técnicos, como problemas de conexão MQTT e falhas nos componentes ESP32 e motores. Após ajustes e substituições, o sistema demonstrou estabilidade e precisão, permitindo a estabilização do pêndulo.

O aplicativo comprovou sua eficácia ao permitir o envio de parâmetros e o monitoramento em tempo real do ângulo do pêndulo, funcionando como uma ferramenta prática para o estudo de controle linear. A solução mostrou-se viável para o controle remoto, com potencial para ser replicada em outros laboratórios, cumprindo seu papel como intermediário entre o aluno e o sistema de controle.

6. BIBLIOGRAFIA

[1] Vasconcelos, Álisson Carvalo & Zanlucchi, José Jean-Paul.: IIoT – Internet das Coisas Industriais. Microsoft Teams – FEMEC42082-1sem2024.

[2] Assis, Pedro Augusto – Relatório de Laboratório 4 – Sistemas instáveis em MA. Microsoft Teams

[3] EICHHORN, Christian. *MQTTnet: .NET library for MQTT-based communication*. [S. l.]: .NET Foundation, 2024. Disponível em: https://www.nuget.org/packages/MQTTnet/. Acesso em: 28 abr. 2025.

7. APÊNDICES

APÊNDICE 1 – Código carregado no ESP-32 – Controlador Kp

```
#include <EWiFi.h>
#include <PubSubClient.h>
#include <WiFiPassword.h>
#include <Encoder.h>

// Pinos do encoder
#define enc_a 22
#define enc_b 23

// Pinos do L298N
#define PWM_PIN 18
#define IN1     19
#define IN2     21

WiFiClient espClient;
PubSubClient MQTT(espClient);

const char* topicSubscribe = "pendulo/parametroskp";
const char* topicPublish = "pendulo/angulokp";

//Definindo outras variaveis uteis
double theta = 0.0, erro = 0.0, u = 0.0, T = 0.0, kp = 0.5;
long contEnc = 0.0;
int count = 0;

// Escolhendo referencia para a velocidade
double thetaRef = 0.0, thetaRefDeg = 0.00; //em radianos

void IRAM_ATTR handleEncoder() {
  static int lastState = LOW;
  int state = digitalRead(enc_a);
  if (state != lastState) {
    if (digitalRead(enc_b) != state) {
      contEnc++;
    } else {
      contEnc--;
    }
  }
  lastState = state;
}

void callback(char* topic, byte* payload, unsigned int length) {
  String mensagemRecebida;
  for (int i = 0; i < length; i++) {
    mensagemRecebida += (char)payload[i];
  }

  Serial.print("Mensagem recebida no tópico ");
  Serial.print(topic);
  Serial.print(": ");
```

```
  Serial.println(mensagemRecebida);

  if (String(topic) == topicSubscribe) {
    // Parse da mensagem recebida (esperando formato: "thetaRefDeg,kp")
    sscanf(mensagemRecebida.c_str(), "%lf,%lf", &thetaRefDeg, &kp);
    Serial.println("Parâmetros atualizados.");
  }
}

void setup() {
  Serial.begin(2000000);
  pinMode(5, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);

  do {
    if(count) Serial.println("\n\nConnection error, trying again.\n");

    count %= 3;

    switch(count) {
      case 0:
        ewifi.setWiFi(SSID1, WPA2_AUTH_PEAP, anonymous, username, userpassword);
        count++;
        break;
      case 1:
        ewifi.setWiFi(SSID2, WPA2_AUTH_PEAP, anonymous, username, userpassword);
        count++;
        break;
      case 2:
        ewifi.setWiFi(SSID3, password);
        count++;
        break;
    }

    ewifi.connect();
  } while(ewifi.status() != WL_CONNECTED);

  count = 0;
  // Setup of Server
  MQTT.setServer(MQTTServer, MQTTPort);

}

void loop() {
  if (!MQTT.connected()) conectarMQTT();
  MQTT.loop();

  noInterrupts();
  long contEncCopy = contEnc;
  interrupts();

  thetaRef = thetaRefDeg*(3.1415/180);

  //Calculando theta a partir da leitura do encoder
```

```
  theta = contEncCopy * 2.0 * 3.1415 / (334.0 * 4.0);

  //Obtendo erro de rastreamento
  erro = thetaRef - theta;

  //Calculando acao do controlador
  T = kp * erro;

  //Alterando sentido de giro de acordo com o sinal do controle
  if (T >= 0){
     //sentido horario
     digitalWrite(7,HIGH);
     digitalWrite(8,LOW);
  }
  else{
     //sentido anti-horario
     digitalWrite(7,LOW);
     digitalWrite(8,HIGH);
     T = -T;
   }

  // Gerando PWM a partir do torque
  u = 19442.0*T + 8.0;

  //Saturando na faixa linear
  u = min(u,100.0);//limitando superiormente
  u = max(u,0.0);//limitando inferiormente

  //Aplicando controle a planta
  analogWrite(PWM_PIN, 255.0 * u / 100.0);

  // Publica o ângulo em graus
  char angleMsg[20];
  snprintf(angleMsg, 20, "%.2f", theta * (180.0/3.1415));
  MQTT.publish(topicPublish, angleMsg);
}

void conectarMQTT() {
  int num = 30;
  char esp_id[num];
  snprintf(esp_id, num, "ESP32-%s", ewifi.getmacAddress());
  while (!MQTT.connected()) {
    if (MQTT.connect(esp_id)) {
      MQTT.subscribe(topicSubscribe);
      Serial.println("Conectado ao broker MQTT.");
    }
  }
}
```

APÊNDICE 2 – Código carregado no ESP-32 – Controlador LEAD

```
#include <EWiFi.h>
#include <PubSubClient.h>
#include <WiFiPassword.h>

// Pinos do encoder
#define enc_a 22
#define enc_b 23
// Pinos do L298N
#define PWM_PIN 18
#define IN1     19
#define IN2     21

WiFiClient espClient;
PubSubClient MQTT(espClient);

const char* topicSubscribe = "pendulo/parametroslead";
const char* topicPublish = "pendulo/angulolead";

// Variáveis de controle
double theta = 0.0, TempoIni = 0.0, erro = 0.0, erro_deriv = 0.0, u = 0.0, T = 0.0;
double a = 0.00, b = 0.00, k = 0.00, td = 0.00;
volatile long contEnc = 0;
double erro_anterior = 0, T_anterior = 0.0, f = 0.0;
double delta_tempo = 0.001;
double thetaRef = 0.0, thetaRefDeg = 0.00;
int count = 0;

void IRAM_ATTR handleEncoder() {
  static int lastState = LOW;
  int state = digitalRead(enc_a);
  if (state != lastState) {
    if (digitalRead(enc_b) != state) {
      contEnc++;
    } else {
      contEnc--;
    }
  }
  lastState = state;
}

void callback(char* topic, byte* payload, unsigned int length) {
  String mensagemRecebida;
  for (int i = 0; i < length; i++) {
    mensagemRecebida += (char)payload[i];
  }

  Serial.print("Mensagem recebida no tópico ");
  Serial.print(topic);
  Serial.print(": ");
  Serial.println(mensagemRecebida);

  if (String(topic) == topicSubscribe) {
```

```
    // Parse da mensagem recebida (esperando formato: "thetaRef,a,b,k,td")
    sscanf(mensagemRecebida.c_str(), "%lf,%lf,%lf,%lf,%lf", &thetaRefDeg, &a, &b, &k, &td);
    Serial.println("Parâmetros atualizados.");
  }
}

void setup() {
  Serial.begin(115200);

  pinMode(PWM_PIN, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(enc_a, INPUT_PULLUP);
  pinMode(enc_b, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(enc_a), handleEncoder, CHANGE);

  // Conectar Wi-Fi com múltiplas redes
  do {
    if (count) Serial.println("\n\nConnection error, trying again.\n");

    count %= 3;

    switch (count) {
      case 0:
        ewifi.setWiFi(SSID1, WPA2_AUTH_PEAP, anonymous, username, userpassword);
        count++;
        break;
      case 1:
        ewifi.setWiFi(SSID2, WPA2_AUTH_PEAP, anonymous, username, userpassword);
        count++;
        break;
      case 2:
        ewifi.setWiFi(SSID3, password);
        count++;
        break;
    }

    ewifi.connect();
  } while (ewifi.status() != WL_CONNECTED);

  count = 0;
  MQTT.setServer(MQTTServer, MQTTPort);
  MQTT.setCallback(callback);
}

void loop() {
  if (!MQTT.connected()) conectarMQTT();
  MQTT.loop();

  TempoIni = micros();

  noInterrupts();
  long contEncCopy = contEnc;
  interrupts();
```

```
  theta = contEncCopy * 2.0 * 3.1415 / (334.0 * 4.0);

  if (TempoIni / 1000000.0 > 8.0) {
    thetaRef = thetaRefDeg * (3.1415 / 180);
  }

  erro = thetaRef - theta;
  erro_deriv = (erro - erro_anterior) / delta_tempo;

  f = -b * T_anterior + k * (erro_deriv + a * erro);
  T = T_anterior + (f * delta_tempo);

  T_anterior = T;
  erro_anterior = erro;

  // Direção
  if (T >= 0) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
  } else {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    T = -T;
  }

  // Saturação
  u = 19442.0 * T + 8.0;
  u = min(u, 100.0);
  u = max(u, 0.0);


  analogWrite(PWM_PIN, 255.0 * u / 100.0);
  Serial.println(u);

  while ((micros() - TempoIni) < 1000.0) { }  // Delay para manter taxa de atualização
  delay(td);

  char angleMsg[20];
  snprintf(angleMsg, 20, "%.2f", theta * (180.0/3.1415));
  MQTT.publish(topicPublish, angleMsg);
}

void conectarMQTT() {
  int num = 30;
  char esp_id[num];
  snprintf(esp_id, num, "ESP32-%s", ewifi.getmacAddress());

  while (!MQTT.connected()) {
    if (MQTT.connect(esp_id)) {
      MQTT.subscribe(topicSubscribe);
      Serial.println("Conectado ao broker MQTT.");
    }
  }
}
```