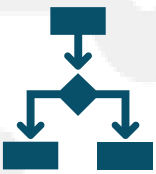
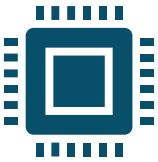
Roberto Higino Pereira da Silva
Joésia Moreira Julião Pacheco
Joelma Monteiro de Carvalho
Organizadores

Curso de Capacitação Profissional Técnica

Hardware e Firmware IoT com IMPC HTNB32L

Módulo 1: Fundamentos de IoT e Sistemas Embarcados


Manaus-AM

2025

EQUIPE TÉCNICA

Diretor:

Raimundo Cláudio Souza Gomes

Coordenador Geral

Roberto Higino Pereira da Silva

Coordenadores Pedagógicos:

Joésia Moreira Julião Pacheco

Joelma Monteiro de Carvalho

Professores

Celso Barbosa Carvalho

Rafael Facioni Scalabrin

Apoio Técnio Hana Electronics

Weslley Fábio Ferreira Santos

SUMÁRIO

APRESENTAÇÃO

O curso de Capacitação Profissional Técnica em Hardware e Firmware IoT com IMCP HTNB32L, 120 horas, tem como objetivo principal capacitar os participantes nos fundamentos da Internet das Coisas (IoT) e sistemas embarcados, promovendo o desenvolvimento de competências técnicas voltadas à integração entre hardware, firmware e plataformas em nuvem.

Ao longo de 120 horas de atividades teóricas e práticas, os alunos serão introduzidos aos principais conceitos de IoT, explorando arquiteturas de sistemas embarcados, tipos de microcontroladores e suas aplicações, além de redes sem fio, protocolos de comunicação e aspectos de segurança em dispositivos conectados.

A formação contempla ainda noções básicas de programação para sistemas embarcados, com foco na integração com sensores, atuadores e serviços de nuvem. Através do uso do microcontrolador IMCP HTNB32L, os alunos desenvolverão projetos práticos que simulam aplicações reais em áreas como automação residencial, monitoramento remoto e agricultura inteligente.

Complementando a capacitação, serão abordados temas atuais do ecossistema de desenvolvimento, como o uso de sistemas de controle de versão (Git), além do estudo de arquiteturas de hardware e software baseadas nos principais padrões industriais e plataformas amplamente adotadas no mercado.

Este curso é uma iniciativa do Lab.4.0 (Hub – Tecnologia e Inovação), no âmbito de extensão universitária da Universidade do Estado do Amazonas (UEA) e visa contribuir para a formação técnica e a inserção profissional em áreas estratégicas e inovadoras do setor tecnológico.

Professor Roberto Higino Pereira da Silva

Coordenador do projeto

Módulo 1: Fundamentos de IoT e Sistemas Embarcados

Neste módulo, serão apresentados os primeiros passos no universo da Internet das Coisas (IoT) e dos Sistemas Embarcados.

Prepare-se para entender como sensores, dispositivos, redes e a nuvem trabalham juntos para criar soluções tecnológicas inovadoras que estão mudando o nosso dia a dia.

Objetivo do Módulo:

Capacitar o aluno nos fundamentos de IoT e sistemas embarcados, abordando:

- Arquitetura de dispositivos IoT
- Protocolos de comunicação
- Programação básica de Hardware
- Integração com sensores e com a nuvem

Carga Horária: 30 horas

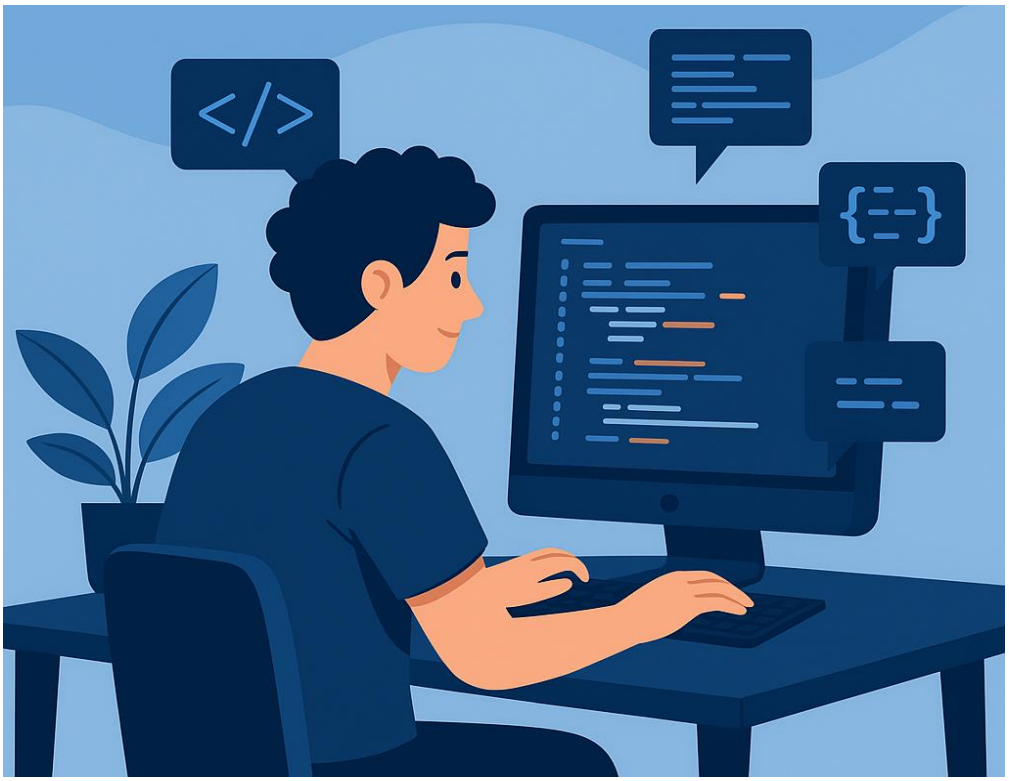
Capítulos deste Módulo:

- Capítulo 1: Fundamentos de IoT e Sistemas Embarcados.
- Capítulo 2: Protocolo MQTT: Comunicação Eficiente para a Internet das Coisas.

Capítulo 1: Fundamentos de IoT e Sistemas Embarcados

Lâmpadas que acendem com a voz, sensores que avisam quando uma planta precisa de água ou máquinas que se comunicam sozinhas para evitar falhas? Neste capítulo, serão explorados os princípios que tornam tudo isso possível — e começar uma jornada pelo fascinante mundo da Internet das Coisas e dos sistemas embarcados.

Figura 1: Fundamentos de IoT e Sistemas Embarcados



Fonte: Lab 4.1 (Equipe técnica do projeto)

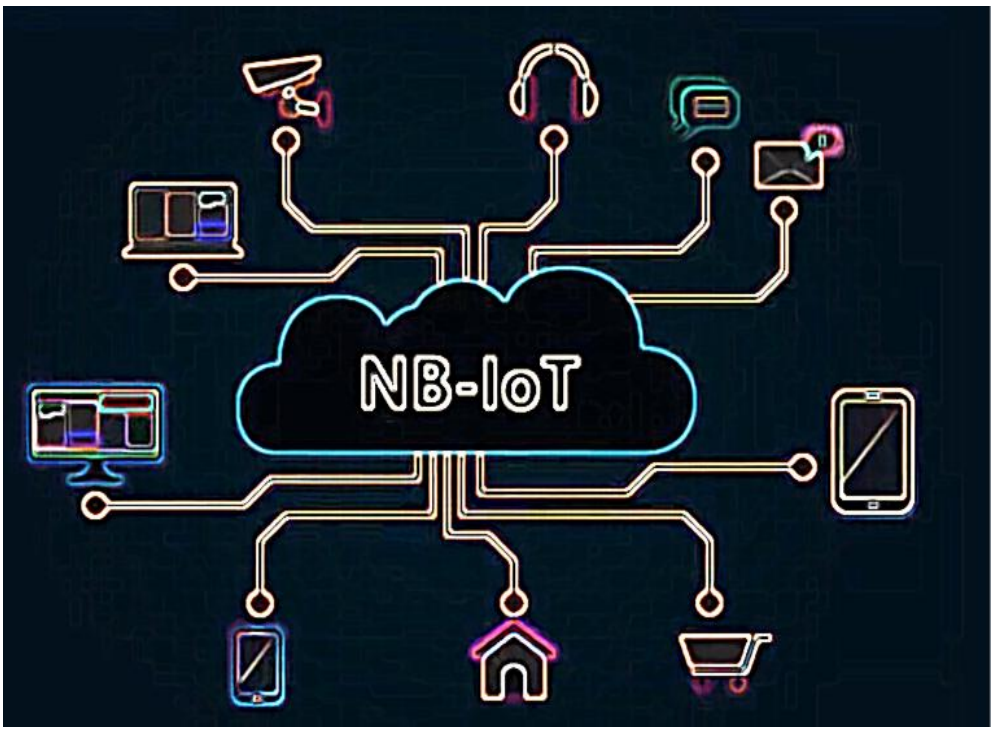
1.Conceitos Iniciais e NB-IoT

IoT – Refere-se à interconexão de dispositivos físicos à internet. Pode-se citar exemplos como: Sensores, atuadores e dispositivos inteligentes que enviam e recebem dados

O NB-IoT (Narrowband IoT) é uma tecnologia de rede de área ampla e baixo consumo de energia (LPWAN – Low Power Wide Area Network), considerada uma das mais promissoras para aplicações em larga escala com dispositivos de sensoriamento.

Essa tecnologia foi desenvolvida com foco na Internet das Coisas (IoT – Internet of Things), permitindo a conexão eficiente de milhares de dispositivos com baixo consumo de energia e longo alcance.

Figura 2: NB-IoT: Sensores, atuadores e dispositivos inteligentes.



Fonte:https://www.teamtweaks.com/blog/wpcontent/uploads/2020/06/banner-5.jpg adaptado de teamtweaks (2020)

A NB- IoT tem diversas aplicações no dia a dia:

1. Automação residencial (luzes, câmeras, controle de temperatura)
2. Cidades inteligentes (monitoramento de tráfego, iluminação pública)
3. Saúde digital (monitoramento de pacientes)
4. Indústria 4.0: Sensores e Manutenção Preditiva

Figura 3: Mundo NB-IoT

Fonte: Lab 4.1 (Equipe técnica do projeto)

2.Sistemas Embarcados

Sistemas embarcados são computadores projetados para realizar funções específicas dentro de um dispositivo maior. Eles são encontrados em produtos do dia a dia, como micro-ondas, carros, máquinas industriais e dispositivos IoT.

Diferente dos computadores comuns, eles geralmente têm hardware e software dedicados e funcionam com baixo consumo de energia, alta eficiência e autonomia.

Os sistemas embarcados podem utilizar tanto microprocessadores quanto microcontroladores, dependendo da complexidade da aplicação:

Figura 4: NB-IoT: Sensores, atuadores e dispositivos inteligentes.

Fonte: Lab 4.1 (Equipe técnica do projeto)

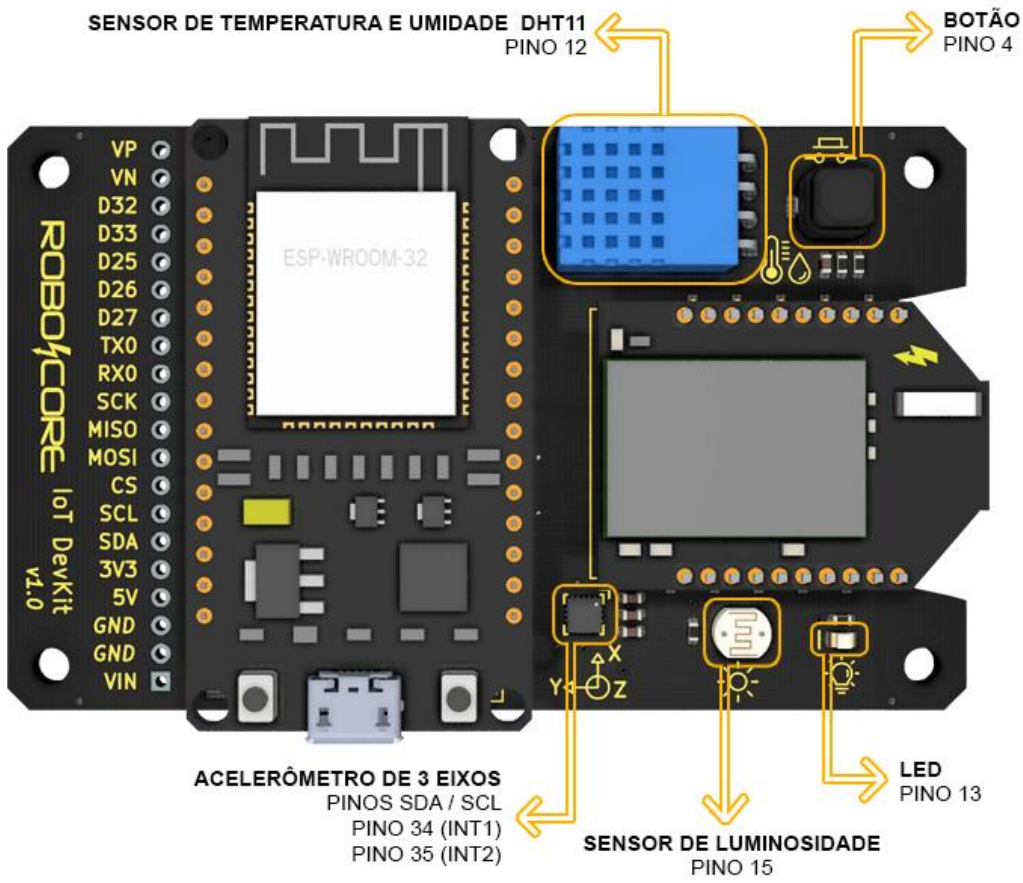
3.Introdução ao IoT DevKit

O IoT DevKit é uma plataforma de desenvolvimento voltada para a criação de protótipos e aplicações em Internet das Coisas (IoT). Essa plataforma reúne os principais recursos necessários para testar e desenvolver soluções conectadas de forma rápida e prática. Nesse módulo, será utilizado o dev kit ilustrado na Figura 5.

Principais Componentes:

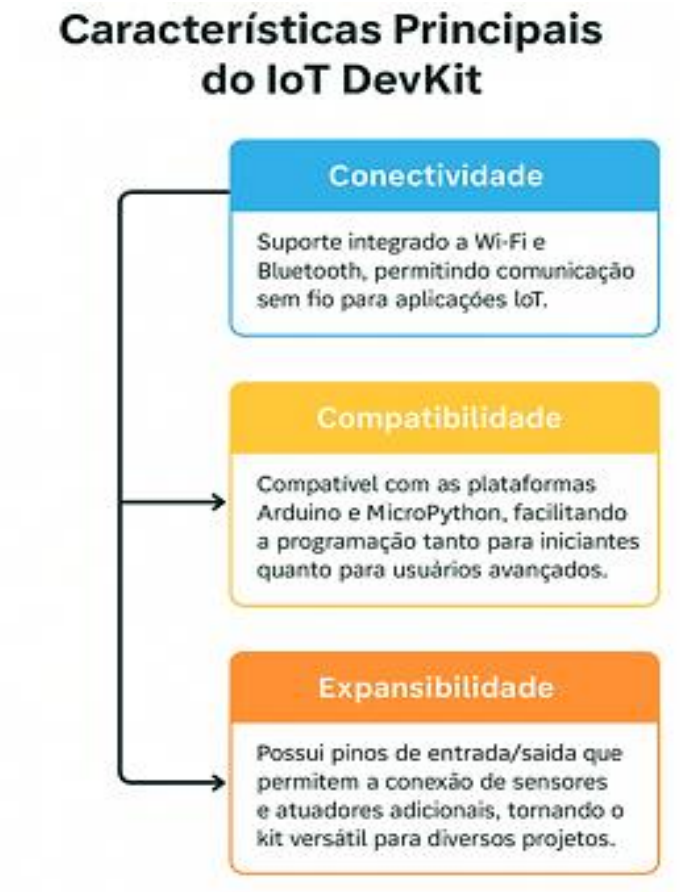
- Placa de desenvolvimento com microcontrolador integrado;
- Sensores embarcados, como temperatura, luminosidade, movimento, entre outros;
- Módulos de comunicação, como Wi-Fi, Bluetooth ou LoRa (ESP-WROOM-32); Ideal para uso educacional, testes de conceito e projetos iniciais em IoT.

Figura 5: Componentes da placa de desenvolvimento



Fonte: https://d229kd5ey79jzj.cloudfront.net/1255/iot-devkit-pinout.png (2023)

Figura 6: Características Principais do Iot DevKit



Fonte: Lab 4.1 (Equipe técnica do projeto)

Recursos e Suporte

- Site oficial: www.robocore.net
- Fórum da comunidade
- Tutoriais e materiais adicionais

4.Configurando o IoT DevKit

Requisitos

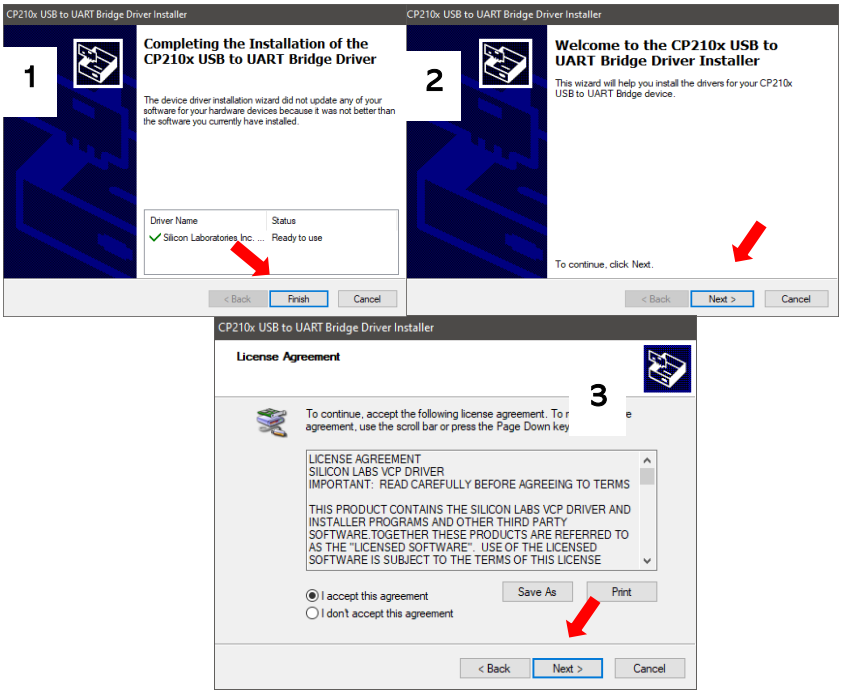
- IoT DevKit
- Cabo USB
- Computador com Windows, macOS ou Linux
- Ambiente de desenvolvimento Arduino IDE ou VS Code.

Passo a Passo

1. Instalação da IDE (Arduino ou PlatformIO)

1.1. Baixe e instale o Arduino IDE em: https://www.arduino.cc/en/software

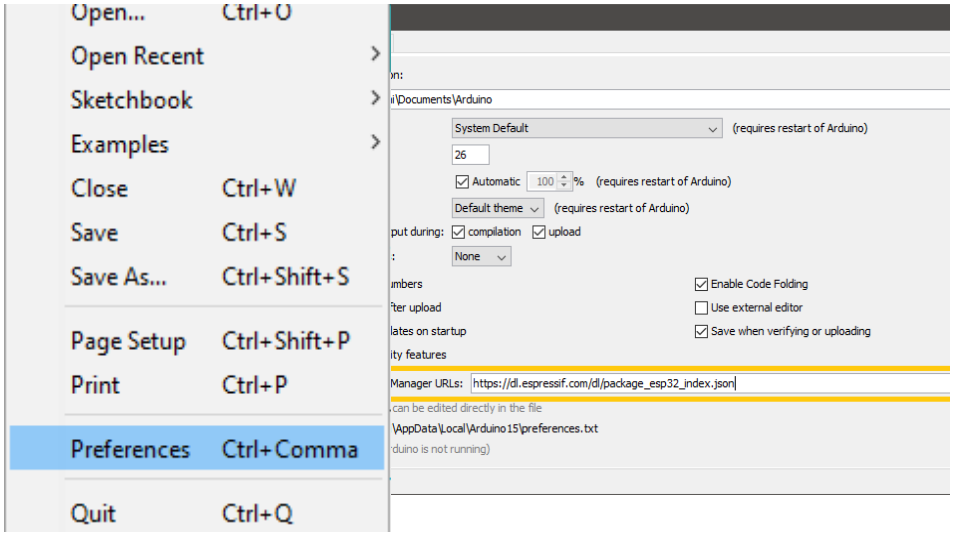
Figura 7: Baixando e Instalando o Arduino IDE



Fonte: Carvalho,2025

1.2. No Arduino IDE, vá em 'Preferências' e adicione a URL do gerenciador de placas: https://dl.espressif.com/dl/package_esp32_index.json ou https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

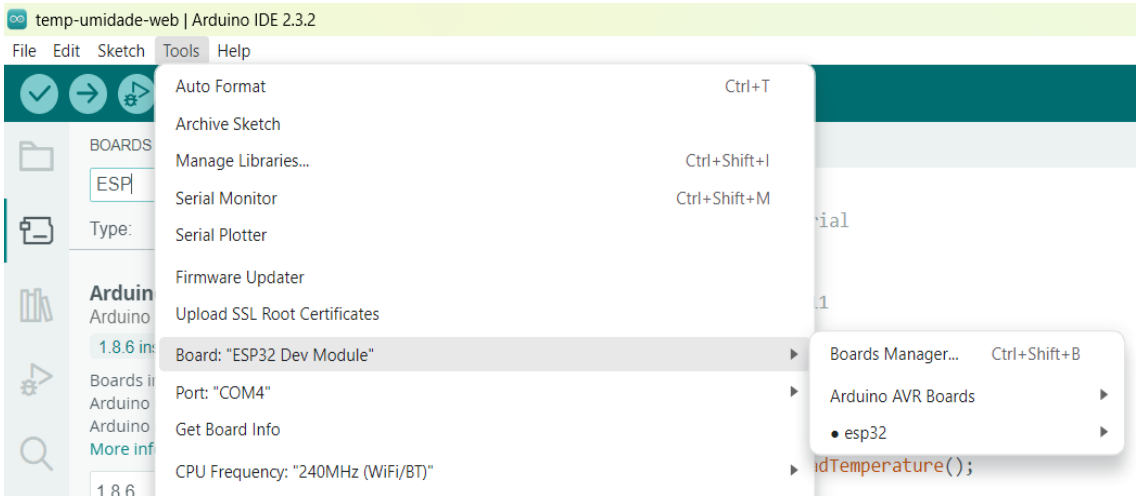
Figura 8: Arduino IDE, vá em 'Preferências' e adicione a URL



Fonte: Carvalho,2025

1.3. Acesse 'Gerenciador de Placas' e instale 'ESP32 by Espressif Systems'.

Figura 9: Acessando 'Gerenciador de Placas' e instale 'ESP32 by Espressif Systems'



Fonte: Carvalho,2025

2. Conectar o kit ao computador

2.1. Conecte o IoT DevKit ao computador via cabo USB;

Figura 10: Conectando o IoT DevKit ao computador via cabo USB. LED ligando

Fonte: https://d229kd5ey79jzj.cloudfront.net/1255/iot-devkit-pinout.png (2023)

2.2. Abra o Gerenciador de Dispositivos (Windows) ou Terminal (Mac/Linux) e verifique a porta serial:

Windows: Dispositivos e Impressoras -> Portas COM

Mac/Linux: Use o comando `ls /dev/tty.*` no terminal.

Resolução de possíveis problemas

Se houver falhas no upload:

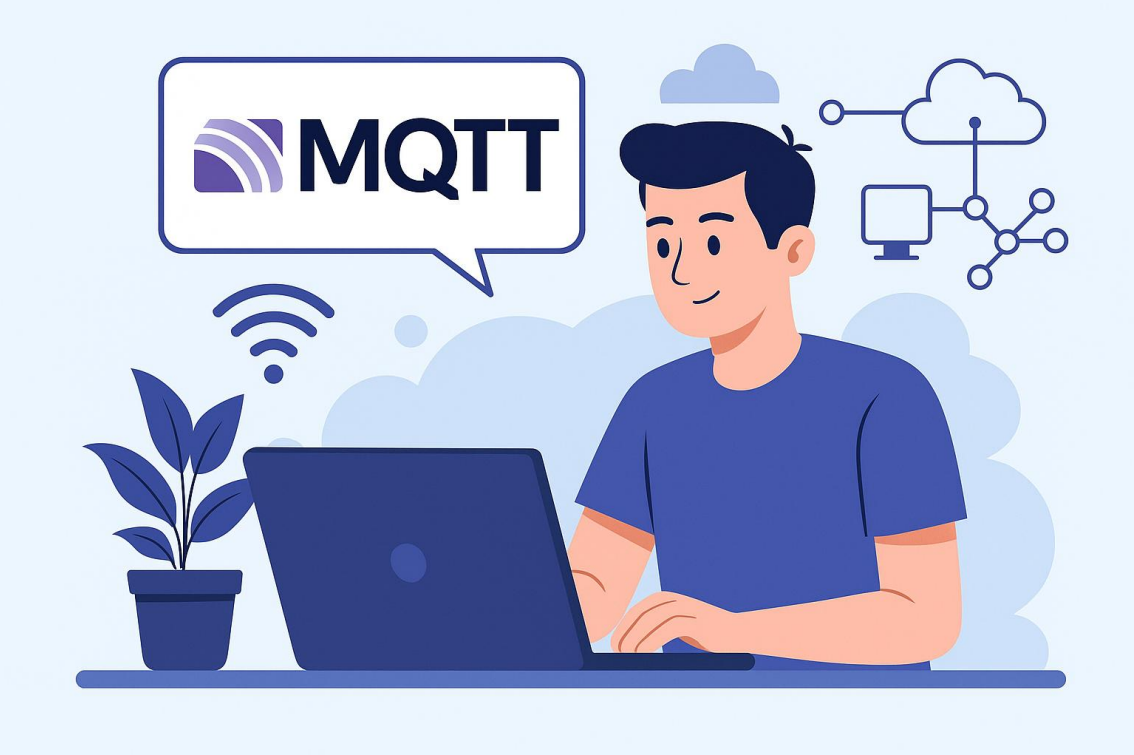
1. Verifique se os drivers USB estão instalados corretamente.
2. Troque a porta USB ou use um cabo diferente.
3. Pressione e segure o botão 'BOOT' enquanto faz o upload.
4. Atualize o firmware da placa, se necessário.

Capítulo 2:Protocolo MQTT - Comunicação Eficiente para a Internet das Coisas

No coração de muitas soluções IoT está o MQTT, um protocolo de comunicação leve, eficiente e ideal para ambientes com recursos limitados.

Nesse capítulo serão explorados os conceitos e aplicações do Protocolo MQTT e suas aplicações.

Figura 11: Conhecendo o MQTT



Fonte: Lab 4.1 (Equipe técnica do projeto)

O MQTT é um protocolo leve de mensagens, baseado no modelo publicação/assinatura.Foi criado pela IBM em 1999 para realizar monitoramento remoto de oleodutos via satélite, sendo projetado para operar em redes com baixa largura de banda, alta latência ou conexões instáveis.

Hoje, o MQTT é um dos protocolos mais utilizados em aplicações de Internet das Coisas (IoT), devido à sua eficiência e simplicidade.

2.Onde o MQTT é utilizado?

O MQTT é amplamente usado em aplicações que envolvem grande quantidade de dispositivos conectados com poucos recursos computacionais, como:

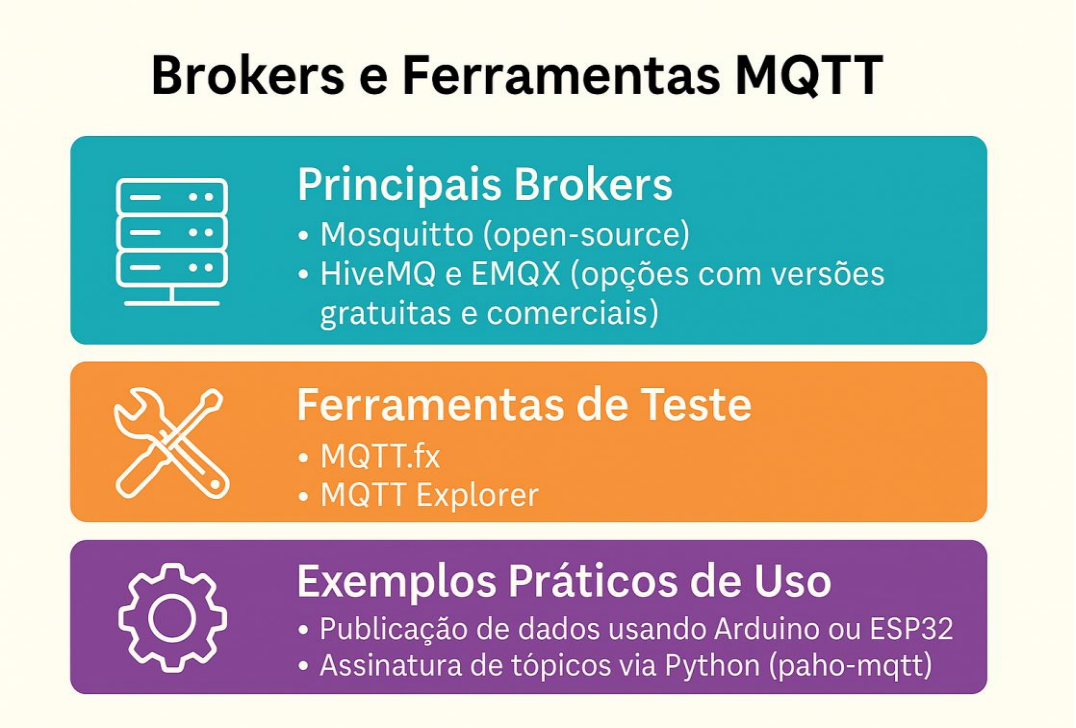
- Sensoriamento remoto (IoT)
- Automação residencial
- Agricultura de precisão
- Logística e rastreamento de ativos

Sua leveza e eficiência o tornam ideal para sistemas que precisam de comunicação rápida e com baixo consumo de energia.

Figura 11: Aplicações do MQTT

Fonte: Lab 4.1 (Equipe técnica do projeto)

Figura 12: Brokers e Ferramentas MQTT



Fonte: Lab 4.1

Hora de praticar!

1. O código fornecido faz um LED piscar em um intervalo fixo de tempo:

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);  // Configura o pino do LED como saída
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // Liga o LED
  delay(1000);                      // Mantém ligado por 1 segundo

  digitalWrite(LED_BUILTIN, LOW);   // Desliga o LED
  delay(59000);                     // Espera até completar 1 minuto (total de 60.000 ms)
}
```

Hora de praticar!

Seu objetivo nos exercícios é:

Modificar o código para alterar o comportamento do LED:

Alterar o intervalo de tempo do delay() para 500ms.

Modificar o código para fazer o LED piscar três vezes rapidamente e depois esperar 1 segundo antes de repetir.

2. Criar uma **função** chamada piscarLED(int vezes, int tempo) que permita definir **quantas vezes o LED pisca e qual o tempo entre os piscas**.

- Dicas
  - A função digitalWrite(pino, HIGH); acende o LED
  - digitalWrite(pino, LOW); apaga o LED.
  - O delay(tempo); define por quanto tempo o LED fica ligado ou desligado.
  - A função for pode ser usada para repetir o efeito de piscar várias vezes.

3. Implementar o Código Morse "SOS" no ESP32

- **Objetivo:**

  Modificar o código para que o LED **pisque no padrão do código Morse para "SOS"**:

- **Código Morse para SOS:**

  S → "•••" (três piscadas curtas)

  O → "———" (três piscadas longas)

  S → "•••" (três piscadas curtas)

- **Repetir o padrão após uma pausa**

Passos:

- Entender o código Morse para "SOS" e converter para piscadas de LED.
- Criar uma função piscarMorse() para controlar o LED no padrão correto.
- Implementar três piscadas curtas (S), três longas (O) e três curtas (S).
- Adicionar um intervalo de 2 segundos antes de repetir a sequência.

Hora de praticar!

Dicas

- Curta (•) → LED ligado por 200ms, desligado por 200ms.
- Longa (–) → LED ligado por 600ms, desligado por 200ms.
- Intervalo entre letras (S e O) → 600ms de pausa.
- Intervalo entre palavras → 2 segundos antes de repetir.

Gabarito

Gabarito da Questão 1:

1. **Alterar o intervalo de tempo para 500ms:**

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // Liga o LED
  delay(500);                       // Espera 500ms
  digitalWrite(LED_BUILTIN, LOW);   // Desliga o LED
  delay(500);                       // Espera 500ms
}
```

2. **Fazer o LED piscar 3 vezes rapidamente com 200ms de intervalo e depois esperar 1 segundo:**

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  for (int i = 0; i < 3; i++) {
    digitalWrite(LED_BUILTIN, HIGH);  // Liga o LED
    delay(200);                       // Espera 200ms
    digitalWrite(LED_BUILTIN, LOW);   // Desliga o LED
    delay(200);                       // Espera 200ms
  }
  delay(1000);  // Espera 1 segundo antes de repetir
}
```

Gabarito

Gabarito da Questão 1:

3. Criar a função piscarLED(int vezes, int tempo):

```
void piscarLED(int vezes, int tempo) {
  for (int i = 0; i < vezes; i++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(tempo);
    digitalWrite(LED_BUILTIN, LOW);
    delay(tempo);
  }
}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  piscarLED(3, 200);   // Pisca 3 vezes com intervalo de 200ms
  delay(1000);         // Espera 1 segundo antes de repetir
```

Gabarito

Gabarito da Questão 2:

```
void piscarLED(int vezes, int tempo) {
  for (int i = 0; i < vezes; i++) {
    digitalWrite(LED_BUILTIN, HIGH);  // Liga o LED
    delay(tempo);                     // Espera o tempo definido
    digitalWrite(LED_BUILTIN, LOW);   // Desliga o LED
    delay(tempo);                     // Espera novamente antes da próxima piscada
  }
}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);       // Configura o LED como saída
}

void loop() {
  piscarLED(5, 300);                  // Exemplo: Piscar o LED 5 vezes com 300ms
de intervalo
  delay(2000);                        // Aguarda 2 segundos antes de repetir
}
```

Gabarito

Explicação:

Função piscar LED:

Recebe dois parâmetros:

- vezes → Quantidade de piscadas
- tempo → Tempo de espera (em milissegundos) entre cada acendimento e apagamento.

Exemplo de uso no loop():

Pisca o LED 5 vezes, com 300ms de intervalo entre os estados, e depois espera 2 segundos antes de repetir.

Gabarito da Questão 3:

```
void piscarCurta() {
  digitalWrite(LED_BUILTIN, HIGH);  // Liga o LED
  delay(200);                       // Mantém aceso por 200ms (curta)
  digitalWrite(LED_BUILTIN, LOW);   // Desliga o LED
  delay(200);                       // Pausa entre os sinais
}

void piscarLonga() {
  digitalWrite(LED_BUILTIN, HIGH);  // Liga o LED
  delay(600);                       // Mantém aceso por 600ms (longa)
  digitalWrite(LED_BUILTIN, LOW);   // Desliga o LED
  delay(200);                       // Pausa entre os sinais
}
```

Gabarito

```
void piscarMorse() {
  // Letra S: três curtas
  for (int i = 0; i < 3; i++) {
    piscarCurta();
  }

  delay(600);  // Pausa entre letras

  // Letra O: três longas
  for (int i = 0; i < 3; i++) {
    piscarLonga();
  }

  delay(600);  // Pausa entre letras

  // Letra S: três curtas novamente
  for (int i = 0; i < 3; i++) {
    piscarCurta();
  }
}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);  // Define o pino do LED como saída
}

void loop() {
  piscarMorse();      // Executa o sinal de SOS
  delay(2000);        // Espera 2 segundos antes de repetir
}
```

Gabarito

Explicação:

- Curta (•): LED ligado por 200ms → desligado por 200ms.
- Longa (–): LED ligado por 600ms → desligado por 200ms.
- Intervalo entre letras (S → O → S): 600ms.
- Intervalo entre as repetições do SOS: 2 segundos (2000ms).

Links Úteis - Explorando a IoT

IoT DevKit - 7. Introdução ao LoRaWAN - Tutoriais - RoboCore

IoT DevKit - 8. Envio de Dados por LoRaWAN - Tutoriais - RoboCore

IoT DevKit - 11. Introdução ao Wi-Fi - Tutoriais - RoboCore

Referências

ESPRESSIF. Documentação oficial da biblioteca WiFi do ESP32 (Arduino). [S. l.], [2020]. Disponível em: https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi. Acesso em: 28 jul. 2025.

ESPRESSIF. Documentação oficial do ESP32. [S. l.], [2020]. Disponível em: https://docs.espressif.com. Acesso em: 28 jul. 2025.

MQTT.ORG. Site oficial do protocolo. [S. l.], [2020]. Disponível em: https://mqtt.org. Acesso em: 28 jul. 2025.

RANDOM NERD TUTORIALS. Tutorial: conectando o ESP32 ao Wi-Fi e realizando uma requisição HTTP. [S. l.], [2020]. Disponível em: https://randomnerdtutorials.com/esp32-http-get-post-arduino/. Acesso em: 28 jul. 2025.

RANDOM NERD TUTORIALS. Tutorial prático MQTT com ESP32 + Mosquitto + MQTT.fx. [S. l.], [20--?]. Disponível em: https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/. Acesso em: 28 jul. 2025.

SCIKIT-LEARN. Documentação do Scikit-learn: biblioteca de machine learning em Python. [S. l.], [2020]. Disponível em: https://scikit-learn.org/stable/. Acesso em: 28 jul. 2025.

THE THINGS NETWORK. Comunidade e plataforma aberta para LoRaWAN. [S. l.], [2020]. Disponível em: https://www.thethingsnetwork.org. Acesso em: 28 jul. 2025.