PETEE UFMG

Universidade Federal de Minas Gerais

Apostila de Internet das Coisas

9 de janeiro de 2025

Sumário

1 Introdução **3**

2 Objetivo da Apostila **3**

3 Medidas de Segurança **3**

4 Placas e Shields **4**
4.1 Placas . . . 4
4.1.1 Arduino Uno . . . 4
4.1.2 Arduino Nano 33 IoT . . . 5
4.1.3 ESP32 . . . 5
4.1.4 ESP8266 . . . 6
4.1.5 Raspberry Pi . . . 6
4.2 Shields . . . 7
4.2.1 Shield com Módulo Lorawan . . . 7
4.2.2 Shield WiFi ESP8266 . . . 8
4.2.3 Shield de Expansão Xbee . . . 8
4.2.4 Shield Data Logger . . . 9
4.2.5 Shield Ethernet W5500 . . . 10

5 Componentes Básicos **10**
5.1 Resistor e Potenciômetro . . . 10
5.2 Capacitor . . . 11
5.3 Indutor . . . 12
5.4 Diodo . . . 12
5.5 Conectores . . . 13
5.6 Botão . . . 14
5.7 Switch . . . 14

6 Componentes de Potência **15**
6.1 Transistor . . . 15
6.2 Regulador de Tensão (LM7805, LM317) . . . 16
6.3 Conversor DC-DC (Step-Up, Step-Down) . . . 16

7 Sensores e Atuadores **17**
7.1 Sensores . . . 17
7.2 Atuadores . . . 19

8 Fontes de Alimentação **20**
8.1 Fontes de tensão 20
8.2 Pilhas 20
8.3 Baterias 20

9 Programação para Dispositivos IoT **21**
9.1 Ferramentas e Linguagens de Programação 21
9.2 Arduino IDE 22
9.2.1 Criando o primeiro arquivo 22
9.2.2 Ferramentas importantes 24
9.3 Estrutura de Programa 27

10 Protocolos de comunicação e segurança **28**
10.1 Protocolos de comunicação 28
10.1.1 Camada de Acesso à Rede 30
10.1.2 Camada de Internet 30
10.1.3 Camada de Transporte 31
10.1.4 Camada de Aplicação 31
10.2 Protocolos de segurança 32
10.2.1 DTLS 32
10.2.2 TLS/SSL 33
10.2.3 OAuth 2.0 34
10.2.4 X.509 Certificates 34

11 Propostas de Projeto **35**
11.1 Ideia de projeto: Plantinha Inteligente 36
11.1.1 Materiais Físicos: 36
11.1.2 Softwares e Interfaces: 36
11.1.3 Linguagens: 36

1 Introdução

A Internet das Coisas, do inglês Internet of Things (IoT), é um termo utilizado para descrever um paradigma tecnológico, no qual os objetos físicos estão conectados em rede e são acessados através da Internet. O termo trás a ideia de que qualquer dispositivo físico ("coisa") pode ser integrado à Internet, desde que esteja conectado a um núcleo de processamento que lhe forneça "inteligência", que nada mais é do que um dispositivo eletrônico que tenha a capacidade de coletar/enviar dados do objeto e apresentá-los em uma interface online. Dessa forma, uma planta, por exemplo, pode estar conectada à internet, uma vez que monitorada por um sistema inteligente, o usuário poderá obter informações e até mesmo dar comandos remotamente ao sistema online ao qual está a planta está inserida.

2 Objetivo da Apostila

Capacitar os participantes para compreender, projetar e implementar soluções de Internet das Coisas (IoT), fornecendo uma visão abrangente dos princípios, tecnologias e aplicações da IoT. Isso inclui a compreensão dos fundamentos da IoT, o desenvolvimento de habilidades práticas em programação de dispositivos IoT, a exploração de casos de uso em diversos setores, o monitoramento e automação de objetos à distância e a atuação em tempo real sobre esses objetos. Além disso, o curso prepara os participantes para aproveitar as oportunidades de carreira nesta área em rápida evolução.

3 Medidas de Segurança

Antes de desenvolver projetos de circuitos elétricos e eletrônicos, é importante adotar algumas medidas de segurança para evitar complicações durante o manuseio dos componentes, como curto-circuitos, choques e queimaduras.

- **Ambiente:** Opte por um ambiente limpo e sem umidade para o desenvolvimento de projetos eletrônicos.
- **Vestimenta:** É preferível usar sapatos fechados e calças compridas enquanto estiver trabalhando com circuitos. Recomenda-se também prender o cabelo e retirar joias e bijuterias.
- **Alimentação:** Enquanto estiver montando e/ou modificando o circuito, mantenha as fontes de alimentação desligadas. Quando finalizar, verifique a montagem e, se possível, meça a continuidade. Conexões erradas podem causar

curto-circuito ou queima de componentes. Além disso, evite tocar no circuito se ele estiver alimentado.

  - Caso use baterias, atente-se às instruções de carregamento.

- **Proteção contra sobrecarga:** Dependendo da quantidade de corrente que fluirá pelo circuito, pode ser interessante instalar fusíveis ou disjuntores, ou incluir um circuito de proteção.

- **Simulação:** Se possível, teste o circuito com ferramentas de simulação, como LTSpice e Tinkercad, antes de realizar a montagem.

4 Placas e Shields

4.1 Placas

4.1.1 Arduino Uno

- **Descrição**: O Arduino Uno é uma das placas mais populares da plataforma Arduino. É uma excelente escolha para iniciantes na área de eletrônica e IoT devido à sua simplicidade e versatilidade.

- **Características**: Equipado com o microcontrolador ATmega328P, possui 14 pinos digitais de entrada/saída (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um cristal oscilador de 16 MHz, uma conexão USB, um conector de alimentação, um cabeçalho ICSP e um botão de reset. Pode ser alimentado via USB ou com uma fonte externa.

- **Aplicações**: Projetos de automação residencial, robótica, controle de motores, sensores de ambiente, entre outros.

Figura 1: Arduino Uno

4.1.2 Arduino Nano 33 IoT

- **Descrição**: O Arduino Nano 33 IoT é uma versão compacta e poderosa da linha Arduino, com conectividade Wi-Fi e Bluetooth integrada, ideal para projetos de IoT.
- **Características**: Utiliza o microcontrolador SAMD21 Cortex-M0+ de 32 bits, possui conectividade Wi-Fi e Bluetooth, 14 pinos digitais de entrada/saída, 8 entradas analógicas, 1 saída DAC, micro USB, interface I2C e SPI, e sensores embutidos.
- **Aplicações**: Automação residencial, dispositivos portáteis, projetos de saúde conectada, sensores ambientais e qualquer aplicação que necessite de conectividade sem fio.

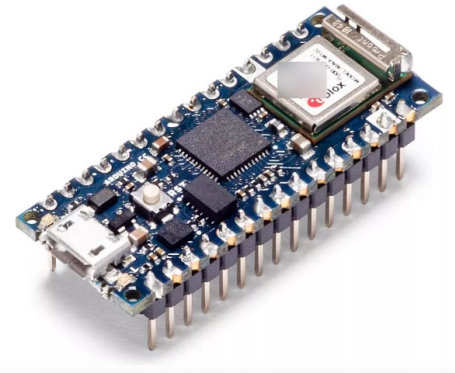

Figura 2: Arduino Nano 33 IoT

4.1.3 ESP32

- **Descrição**: O ESP32 é um microcontrolador versátil com conectividade Wi-Fi e Bluetooth integrada, amplamente utilizado em projetos de IoT devido ao seu desempenho e baixo custo.
- **Características**: Possui um processador dual-core Tensilica LX6, suporte para Wi-Fi e Bluetooth (BLE), diversos pinos de GPIO, ADCs (Conversores Analógico-Digitais), DACs (Conversores Digital-Analógicos), PWM, SPI, I2C e interfaces UART. Inclui também funcionalidades de economia de energia, tornando-o ideal para dispositivos alimentados por bateria.
- **Aplicações**: Automação residencial, dispositivos vestíveis, sensores inteligentes, redes de sensores sem fio, projetos de robótica e mais.

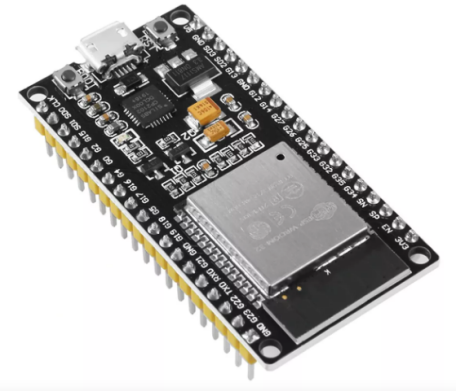

Figura 3: ESP32

4.1.4 ESP8266

- **Descrição**: O ESP8266 é um microcontrolador com Wi-Fi integrado, muito popular em projetos de IoT devido ao seu baixo custo e facilidade de uso.
- **Características**: Equipado com o processador Tensilica Xtensa L106, suporte para Wi-Fi, GPIOs, ADC, SPI, I2C e UART. Possui memória flash integrada que permite armazenar código e dados.
- **Aplicações**: Automação residencial, dispositivos conectados, controle remoto via internet, monitoramento de sensores e outras aplicações que necessitem de conectividade Wi-Fi.

Figura 4: ESP8266

4.1.5 Raspberry Pi

- **Descrição**: O Raspberry Pi é um computador de placa única que oferece capacidades completas de computação. É amplamente utilizado em projetos de IoT, automação, aprendizado de programação e muito mais.

- **Características**: Varia conforme o modelo. O Raspberry Pi 4, por exemplo, possui um processador quad-core ARM Cortex-A72, opções de 2GB, 4GB ou 8GB de RAM, múltiplas portas USB 3.0 e 2.0, Ethernet Gigabit, Wi-Fi, Bluetooth, GPIOs, saídas HDMI e suporte para dois monitores 4K.
- **Aplicações**: Servidores domésticos, media centers, sistemas de automação, projetos de aprendizado de máquina, controle de robótica, entre outros.

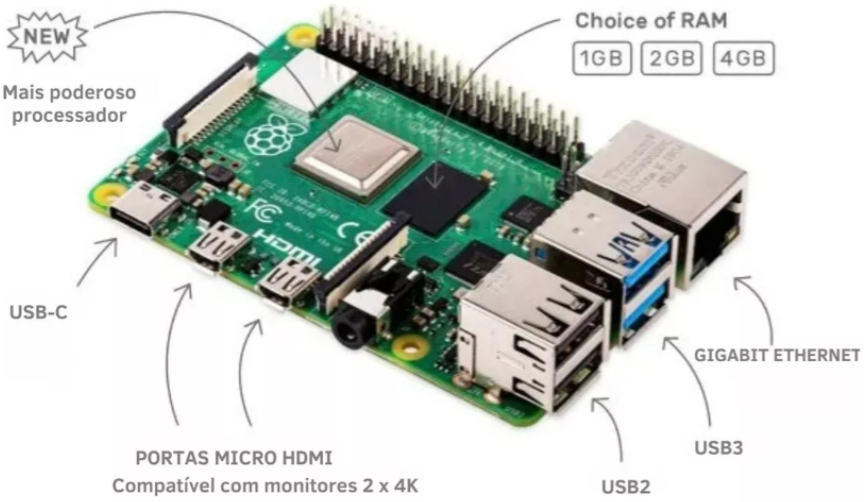

Figura 5: Raspberry Pi

4.2 Shields

4.2.1 Shield com Módulo Lorawan

- **Descrição**: Este shield permite a comunicação em redes LoRaWAN, uma tecnologia de comunicação sem fio de longa distância e baixo consumo de energia, ideal para IoT.
- **Características**: Suporta diversas frequências de operação LoRaWAN, integração fácil com placas Arduino, conector de antena externa e capacidade de transmitir dados a longas distâncias com baixo consumo de energia.
- **Aplicações**: Monitoramento agrícola, cidades inteligentes, monitoramento ambiental, redes de sensores remotos, entre outros.

Figura 6: Shield com Módulo Lorawan

4.2.2 Shield WiFi ESP8266

- **Descrição**: Este shield adiciona conectividade Wi-Fi a placas como o Arduino, facilitando a criação de projetos conectados à internet.
- **Características**: Baseado no módulo ESP8266, suporta protocolos Wi-Fi, interface SPI para comunicação com a placa base, fácil integração com a IDE Arduino.
- **Aplicações**: Automação residencial, projetos de IoT, monitoramento remoto, controle de dispositivos via internet, entre outros.

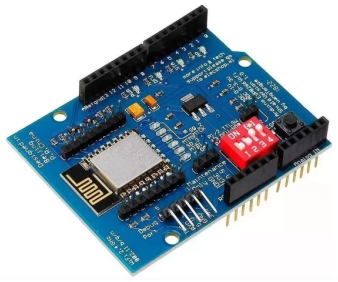

Figura 7: Shield WiFi ESP8266

4.2.3 Shield de Expansão Xbee

- **Descrição**: Facilita a comunicação sem fio usando módulos Xbee, ideais para criar redes de sensores e controle sem fio.
- **Características**: Suporta módulos Xbee, interface fácil de usar com Arduino, LEDs indicadores de status, suporte para múltiplos protocolos de comunicação Xbee.

- **Aplicações**: Redes de sensores sem fio, automação residencial, monitoramento remoto, controle de dispositivos a distância.

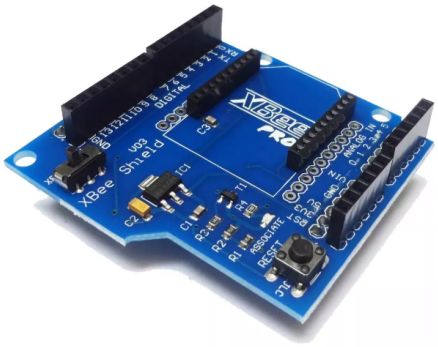

Figura 8: Shield de Expansão Xbee

4.2.4 Shield Data Logger

- **Descrição**: Usado para registrar dados de sensores em um cartão SD, ideal para projetos que necessitam de armazenamento de dados.
- **Características**: Inclui um leitor de cartão SD, RTC (Real-Time Clock) para registro preciso de tempo, fácil integração com a IDE Arduino.
- **Aplicações**: Monitoramento ambiental, registro de dados de temperatura e umidade, projetos de coleta de dados em campo.

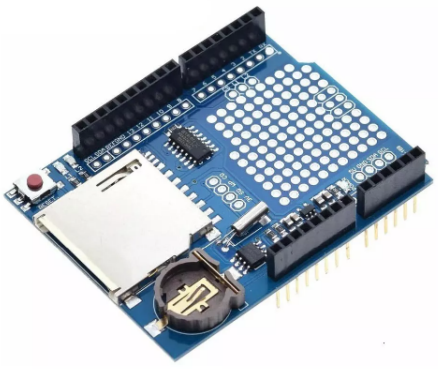

Figura 9: Shield Data Logger

4.2.5 Shield Ethernet W5500

- **Descrição**: Adiciona conectividade Ethernet às placas Arduino, permitindo a conexão com redes locais e internet.
- **Características**: Baseado no chip W5500, suporte para TCP/IP, interface SPI para comunicação com a placa base, LEDs indicadores de status.
- **Aplicações**: Automação residencial, projetos de IoT, servidores web locais, monitoramento remoto, controle de dispositivos via Ethernet.

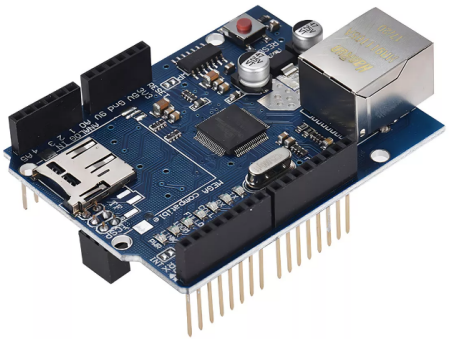

Figura 10: Shield Ethernet W5500

5 Componentes Básicos

5.1 Resistor e Potenciômetro

- **Descrição**: O resistor é um componente eletrônico que limita a corrente elétrica em um circuito. O potenciômetro é um tipo de resistor variável que pode ser ajustado manualmente para alterar a resistência.
- **Características**: Resistores vêm em diversos valores de resistência, medidos em ohms (Ω). Potenciômetros possuem três terminais e um eixo ajustável.
- **Aplicações**: Resistores são usados para controlar a corrente, dividir tensões e proteger componentes eletrônicos. Potenciômetros são usados em controles de volume, ajustes de brilho e calibração de sensores.

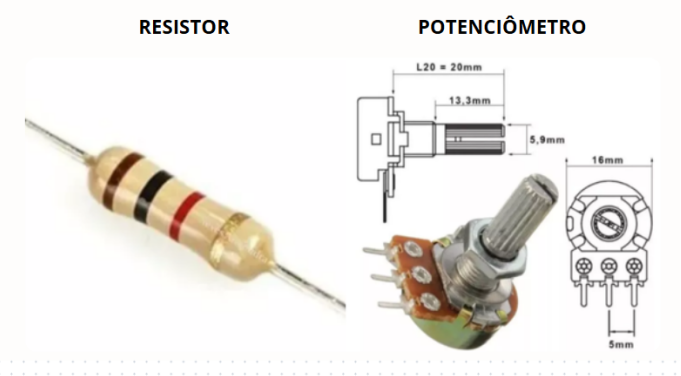

Figura 11: Resistor e Potenciômetro

5.2 Capacitor

- **Descrição**: O capacitor é um componente que armazena energia elétrica em um campo elétrico, podendo liberar essa energia quando necessário.
- **Características**: Capacitores são caracterizados por sua capacitância, medida em farads (F), e pela sua tensão máxima de operação.
- **Aplicações**: São usados para filtragem de ruído, acoplamento e desacoplamento de sinais, armazenamento de energia e temporização em circuitos eletrônicos.

Figura 12: Capacitor

5.3 Indutor

- **Descrição**: O indutor é um componente que armazena energia em um campo magnético quando a corrente elétrica passa através dele.
- **Características**: Indutores são caracterizados pela sua indutância, medida em henrys (H), e pela sua capacidade de corrente.
- **Aplicações**: Usados em filtros, transformadores, fontes de alimentação e em aplicações de armazenamento de energia.

Figura 13: Indutor

5.4 Diodo

- **Descrição**: O diodo é um componente que permite a passagem da corrente elétrica em um único sentido, bloqueando-a no sentido contrário.
- **Características**: Diodos são caracterizados pela sua tensão de condução e pela corrente máxima que podem suportar.
- **Aplicações**: Usados em retificação de corrente alternada para corrente contínua, proteção de circuitos e em circuitos de modulação.

Figura 14: Diodo

5.5 Conectores

- **Descrição**: Jumpers são componentes usados para conectar eletricamente e mecanicamente dois ou mais componentes ou dispositivos.
- **Características**: Os jumpers vêm em diversas formas e tamanhos, como macho-macho, fêmea-fêmea e macho-fêmea, adaptando-se a diferentes tipos de conexões.
- **Aplicações**: Usados em praticamente todos os dispositivos eletrônicos para interconectar placas de circuito, componentes e periféricos.

Figura 15: Conectores

5.6 Botão

- **Descrição**: Botões são dispositivos mecânicos que fecham ou abrem um circuito elétrico quando pressionados.
- **Características**: Vêm em diversas formas e tamanhos, com diferentes tipos de atuação como momentâneo ou travado.
- **Aplicações**: Usados como interruptores de entrada em dispositivos eletrônicos, controles manuais, e em sistemas de reinicialização.

Figura 16: Botão

5.7 Switch

- **Descrição**: Switches são dispositivos que podem interromper a corrente elétrica ou redirecioná-la de um condutor a outro.
- **Características**: Vêm em diversos tipos, incluindo switches de alavanca, push-button, rotativos, e deslizantes, com diferentes configurações de pólo e posição.
- **Aplicações**: Usados em uma ampla gama de aplicações para controlar o fluxo de eletricidade em dispositivos eletrônicos e sistemas elétricos.

Figura 17: Switch

6 Componentes de Potência

6.1 Transistor

- **Descrição**: O transistor é um componente semicondutor utilizado para amplificação e comutação de sinais eletrônicos. Existem diversos tipos de transistores, incluindo bipolares (BJT) e de efeito de campo (FET).
- **Características**: Transistores possuem três terminais: coletor, emissor e base (BJT) ou dreno, fonte e gate (FET). Eles podem operar como interruptores ou amplificadores de corrente.
- **Aplicações**: Amplificação de sinais em circuitos de áudio e rádio, chaveamento em fontes de alimentação e controle de motores.

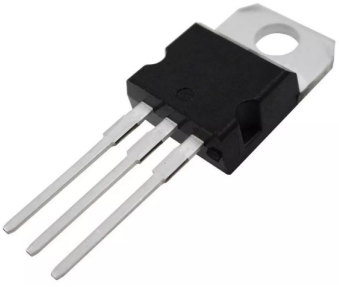

Figura 18: Transistor

6.2 Regulador de Tensão (LM7805, LM317)

- **Descrição**: Reguladores de tensão são componentes que mantêm uma tensão de saída constante independentemente das variações na tensão de entrada ou na carga. Os reguladores LM7805 e LM317 são muito populares.
- **Características**:
  - **LM7805**: Regulador de tensão fixa que fornece 5V na saída. Possui proteção contra curto-circuito e superaquecimento.
  - **LM317**: Regulador de tensão ajustável que pode fornecer uma saída de 1.25V a 37V. Também possui proteção contra curto-circuito e superaquecimento.
- **Aplicações**: Fontes de alimentação para circuitos eletrônicos, estabilização de tensão em sistemas sensíveis, carregadores de bateria.

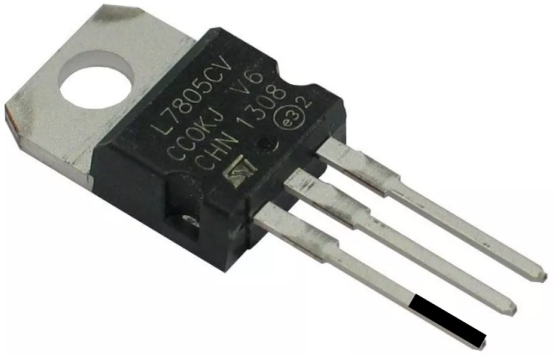

Figura 19: Regulador de Tensão (LM7805, LM317)

6.3 Conversor DC-DC (Step-Up, Step-Down)

- **Descrição**: Conversores DC-DC são circuitos que convertem uma tensão de entrada DC em uma tensão de saída DC diferente. Eles podem aumentar (step-up) ou diminuir (step-down) a tensão.
- **Características**:
  - **Step-Up (Boost)**: Aumenta a tensão de entrada para um nível superior. Usado quando a fonte de alimentação precisa fornecer uma tensão maior que a disponível.

  - **Step-Down (Buck)**: Reduz a tensão de entrada para um nível inferior. Usado para alimentar componentes que requerem uma tensão mais baixa.

- **Aplicações**: Fontes de alimentação de dispositivos portáteis, sistemas de energia renovável, carregadores de bateria, reguladores de tensão em sistemas embarcados.

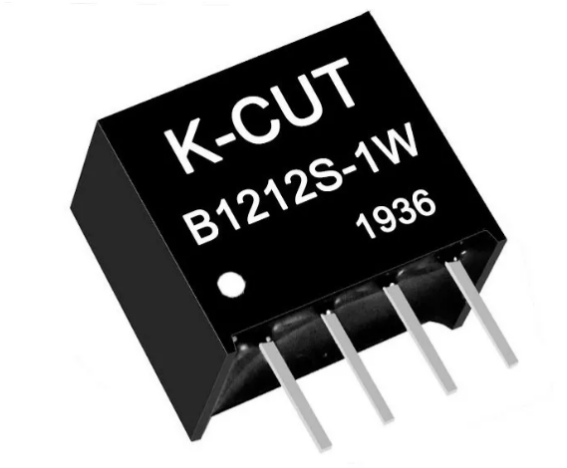

Figura 20: Conversor DC-DC (Step-Up, Step-Down)

7 Sensores e Atuadores

Sensores e atuadores são dispositivos fundamentais em sistemas de automação, controle e Internet das Coisas (IoT). Eles são utilizados para interagir das mais variadas maneiras com o ambiente físico, seja captando sinais ou executando ações no meio físico.

7.1 Sensores

São dispositivos especializados em captar fenômenos físicos do meio ambiente e converter em sinais elétricos para processamento e leitura digital. A seguir, veremos alguns exemplos de sensores.

- **Temperatura**

  - Sensores de temperatura são utilizados para medir a temperatura ambiente ou de objetos específicos. Exemplos incluem termistores, termopares e sensores de temperatura de infravermelho.

- **Umidade**
  - Sensores de umidade são utilizados para medir a quantidade de vapor de água no ar. Exemplos comuns são os higrômetros e sensores capacitivos de umidade.
- **Luz**
  - Sensores de luz detectam a intensidade luminosa. Fotodiodos, fototransistores e LDRs (Light Dependent Resistors) são exemplos de sensores de luz.
- **Pressão**
  - Sensores de pressão medem a força exercida por gases ou líquidos. Barômetros e sensores piezoelétricos são exemplos típicos.
- **Câmeras**
  - Utilizadas para captura de imagens e vídeos. Tipos comuns incluem câmeras CCD e CMOS, que são usadas em diversas aplicações, desde vigilância até reconhecimento de padrões.
- **Movimento**
  - Sensores de movimento detectam alterações no ambiente, como o movimento de pessoas ou objetos. Exemplos incluem sensores PIR (Passive Infrared), acelerômetros e giroscópios.
- **Gás**
  - Sensores de gás detectam a presença de gases específicos no ambiente. Sensores de MQ (e.g., MQ-2, MQ-7) são comuns para detectar gases como metano, propano e monóxido de carbono.
- **Proximidade**
  - Sensores de proximidade detectam a presença de objetos próximos sem contato físico. Exemplos incluem sensores ultrassônicos e sensores de infravermelho.
- **Som**
  - Sensores de som captam ondas sonoras. Microfones são os sensores mais comuns utilizados para detectar som.

- **Encoder**
  - Encoders são utilizados para medir a posição ou velocidade de um eixo rotativo. Encoders ópticos e magnéticos são os tipos mais comuns.

7.2 Atuadores

Os atuadores fazem o caminho inverso dos sensores, eles recebem um sinal digital e com isso conseguem realizar alterações no meio ambiente.

- **Relé**
  - Relés são interruptores eletromecânicos utilizados para controlar circuitos de alta potência com sinais de baixa potência.
- **Motores**
  - Motores são dispositivos que convertem energia elétrica em movimento. Tipos comuns incluem motores de servo, motores de passo e motores DC.
- **LED**
  - LEDs (Light Emitting Diodes) são usados para iluminação e sinalização. Tipos comuns incluem LEDs RGB, que podem emitir várias cores, e LEDs infravermelhos, usados em controles remotos.
- **Display**
  - Displays são utilizados para mostrar informações visuais. Exemplos incluem displays de 7 segmentos, LCDs e OLEDs.
- **Buzzer**
  - Buzzers são dispositivos que produzem som quando ativados, usados frequentemente para alarmes ou notificações.
- **Piezoelétrico**
  - Atuadores piezoelétricos utilizam a propriedade piezoelétrica de certos materiais para criar movimento ou gerar som.

8 Fontes de Alimentação

Para que os projetos IoT possam funcionar, os microcontroladores, sensores, atuadores e outros componentes precisam ser alimentados. Isso significa que eles necessitam receber uma tensão mínima para operarem. A escolha da fonte de alimentação é crucial, pois afeta a eficiência, autonomia e até a portabilidade do projeto. Vamos explorar algumas das opções mais comuns de fontes de energia.

Qual escolher?

8.1 Fontes de tensão

As fontes de tensão são dispositivos que fornecem uma tensão elétrica estável, essencial para o funcionamento adequado de circuitos eletrônicos. Elas podem ser adaptadores de tomada, power banks ou até sistemas mais complexos como painéis solares com reguladores de tensão.

Imagine um pequeno projeto de irrigação automatizada para plantas em sua casa. Conectar um adaptador de tomada ao sistema pode garantir que a bomba de água e os sensores de umidade operem continuamente sem preocupação com o tempo de operação da bateria. Por outro lado, se você deseja que o sistema seja independente da rede elétrica, pode optar por um painel solar.

8.2 Pilhas

As pilhas são uma solução prática e portátil para alimentar pequenos projetos. Elas são fáceis de encontrar, possuem diferentes tamanhos (AA, AAA, botão, etc.) e oferecem uma tensão fixa dependendo do tipo (geralmente 1,5 V para pilhas alcalinas).

Por exemplo, imagine que você está criando um robô movido a pilhas para uma feira de ciências. As pilhas são ideais porque permitem que o robô se mova livremente, sem precisar estar conectado a uma fonte fixa. Além disso, combinando várias pilhas em série, é possível alcançar tensões maiores, como 6 V ao usar quatro pilhas AA.

8.3 Baterias

As baterias são uma alternativa recarregável às pilhas e são amplamente usadas em dispositivos IoT por sua versatilidade e capacidade de armazenamento de energia. Alguns exemplos incluem baterias de íon-lítio (Li-ion) e polímero de lítio (Li-Po), que oferecem alta densidade de energia e são ideais para projetos compactos.

Imagine um drone equipado com sensores de câmera e temperatura. Ele precisa de uma bateria leve, mas com alta capacidade para permitir voos longos. Nesse caso,

uma bateria Li-Po é perfeita, pois equilibra peso e desempenho. Além disso, muitas baterias modernas possuem sistemas de proteção contra sobrecarga, prolongando a vida útil do dispositivo.

—

Escolher a fonte de alimentação correta é como escolher o motor de um carro: deve ser adequado ao tamanho e às necessidades do projeto. Cada tipo de fonte de energia tem suas vantagens e limitações, mas com criatividade e planejamento, é possível criar soluções IoT eficientes e inovadoras.

9 Programação para Dispositivos IoT

A programação é um elemento fundamental para dispositivos IoT, pois possibilita a operação eficiente de sensores, atuadores e controladores. Programar para dispositivos IoT requer atenção especial às limitações dos dispositivos embarcados, como capacidade de memória, processamento e consumo energético, além de lidar com comunicações entre redes heterogêneas. Dessa forma, o software deve levar em conta as seguintes especificações:

- Eficiência: Muitos dispositivos IoT, como microcontroladores, operam com recursos limitados, exigindo códigos otimizados para desempenho e economia de energia.
- Conectividade: A troca de informações entre dispositivos e servidores é essencial, utilizando protocolos como MQTT e HTTP para comunicação em rede.
- Tempo Real: Em muitos casos, dispositivos IoT precisam reagir a eventos de maneira imediata, demandando sistemas em tempo real.
- Escalabilidade: Sistemas IoT frequentemente incluem centenas ou milhares de dispositivos, requerendo soluções que suportem essa complexidade sem comprometer a funcionalidade.

Assim, a escolha das ferramentas de desenvolvimento, como linguagens de programação, bibliotecas e plataformas, deve ser feita com critério, garantindo compatibilidade com as restrições de hardware e atendendo às exigências de desempenho e conectividade do sistema IoT.

9.1 Ferramentas e Linguagens de Programação

Como dito anteriormente, as ferramentas e linguagens de programação utilizadas em projetos IoT variam de acordo com as capacidades do hardware e os objetivos

da aplicação. A escolha deve priorizar soluções que maximizem a compatibilidade com os dispositivos e facilitem a integração com os sistemas em rede. Algumas das opções mais populares incluem:

- C e C++: Ideais para dispositivos embarcados, como ESP32, STM32 e Arduino, por oferecerem alto desempenho e controle sobre o hardware.
- Python: Amplamente utilizado em prototipagem e dispositivos mais robustos, como o Raspberry Pi, graças à sua simplicidade e bibliotecas prontas para IoT.
- JavaScript (Node.js): Comumente usado para desenvolver servidores e aplicações web que interagem com dispositivos IoT.
- Lua: Escolhida para sistemas leves, como firmwares do ESP8266 utilizando NodeMCU.

Entre os ambientes de desenvolvimento mais populares, destacam-se o Arduino IDE e o PlatformIO, que oferecem suporte a uma ampla gama de dispositivos. Frameworks como MicroPython e Node-RED também são amplamente usados para simplificar o desenvolvimento IoT. Nesta apostila será apresentado um pouco de como a Arduino IDE funciona e como o usar para desenvolver seus projetos.

9.2 Arduino IDE

O Arduino IDE (Integrated Development Environment) é o ambiente de desenvolvimento oficial para programação de placas Arduino e outros microcontroladores compatíveis como o ESP32 e o ESP8266. Ele oferece uma interface amigável e acessível, permitindo que desenvolvedores, desde iniciantes até experientes, escrevam, testem e carreguem códigos nos dispositivos. Nesta seção será apresentado um pouco do seu funcionamento. Na imagem abaixo é mostrada a interface inicial do Arduino IDE, na qual são observadas algumas ferramentas e recursos do software.

9.2.1 Criando o primeiro arquivo

Para criar um arquivo no software, basta clicar em "Arquivo"e logo depois em "Novo Esboço", ou utilizar o atalho de teclado **Ctrl+N**.

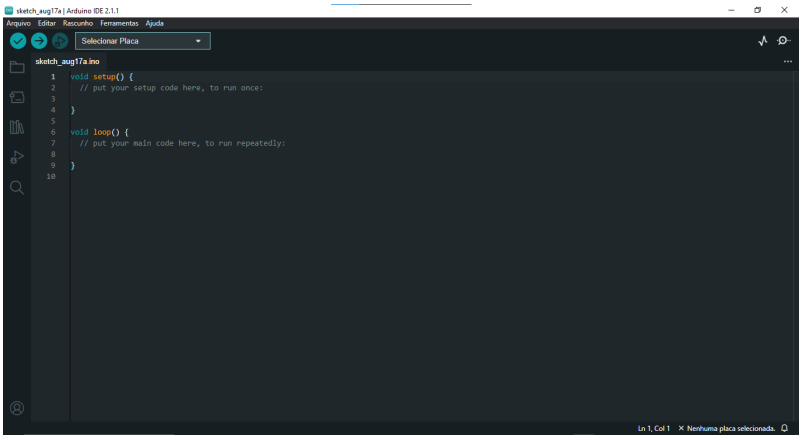

Figura 21: Interface do Arduino IDE

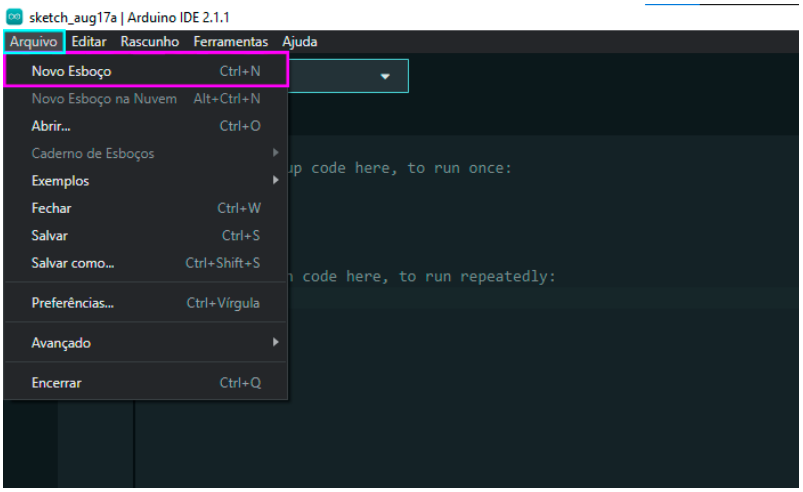

Figura 22: Criando um novo arquivo

Logo após a criação do arquivo é preciso salvá-lo para que as alterações não sejam perdidas, para isso basta clicar no botão mostrado abaixo ou usar o atalho **Ctrl+S** e escolher a pasta na qual o arquivo será salvo.

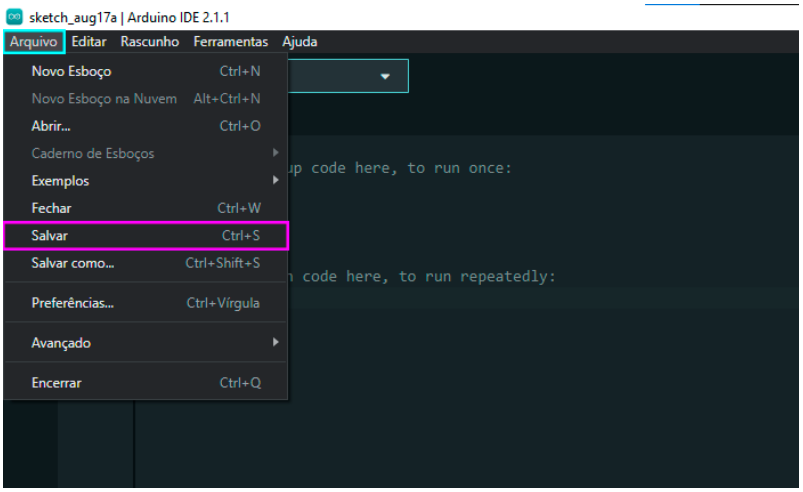

Figura 23: Salvando arquivo no Computador

9.2.2 Ferramentas importantes

Dentro da IDE existem diversas ferramentas, cada uma com sua respectiva utilidade. Abaixo estão listadas algumas das principais e mais recorrentes no uso do software.

Bibliotecas

As bibliotecas são conjuntos de código pré-escrito e podem ser incluídas nos programas de modo a simplificar o desenvolvimento. São necessárias para um bom funcionamento do código, pois alguns comandos comumente utilizados estão configurados em alguma biblioteca específica. É possível acessá-las no Arduino IDE, como mostra a imagem abaixo.

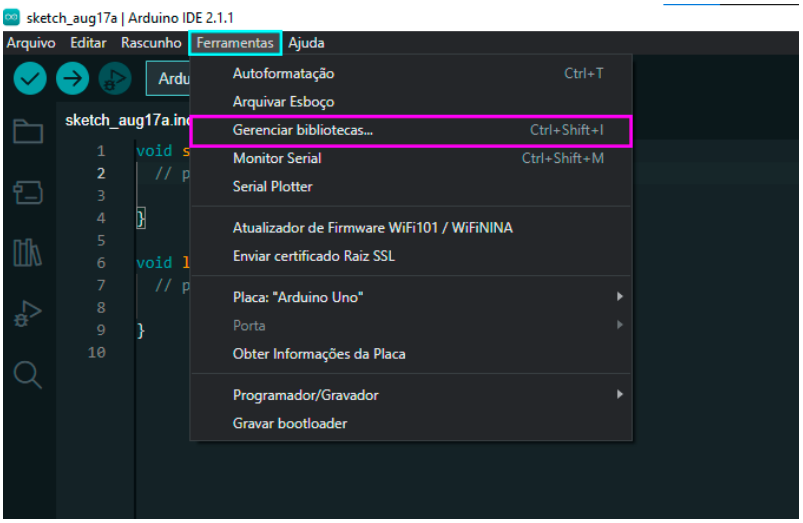

Figura 24: Acessando as bibliotecas na Interface do Arduino

Monitor Serial

O monitor serial é uma ferramenta que permite a visualização de dados e informações gerados pelo microcontrolador durante a execução do código. Existem diversas aplicações nas quais o programa utiliza o monitor serial, pois a partir dele é possível imprimir valores de variáveis, resultados de operações e outras mensagens. A imagem abaixo mostra como ativar a sua visualização.

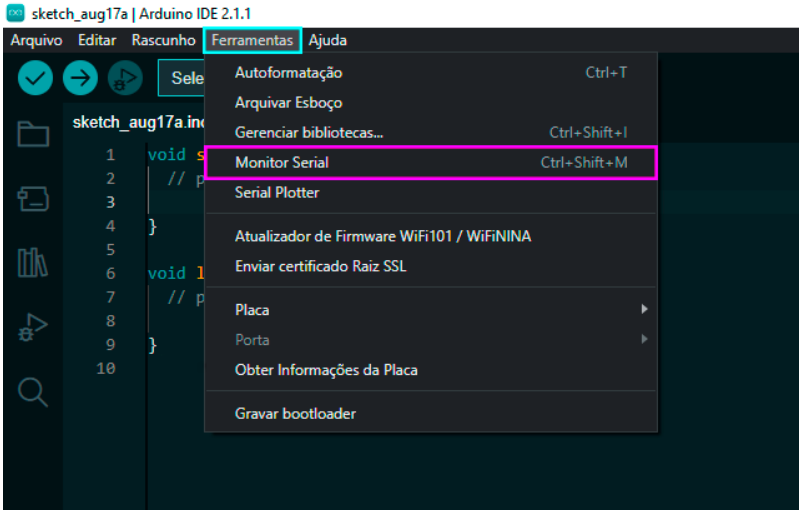

Figura 25: Acessando o Monitor Serial

Seleção da placa e porta

Para que o código possa ser enviado e funcione corretamente é necessário especificar o tipo da placa usada e a porta na qual foi feita a conexão do microcontrolador com o do computador. Abaixo é mostrado como acessar as configurações de Placa e Porta.

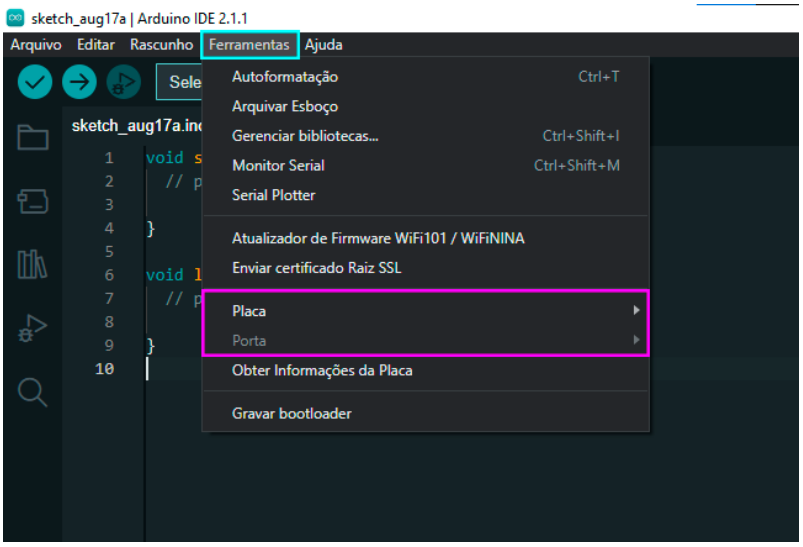

Figura 26: Seleção de Placa e Porta

Executando e compilando o código

O processo de desenvolvimento de software é um ciclo que envolve várias etapas além da escrita do código, como a verificação de possíveis erros e a execução. Isso é especialmente relevante no contexto da programação para dispositivos como o microcontrolador. Para essa finalidade, algumas ferramentas e recursos são fundamentais:

Verificar: Essa etapa envolve a análise da sintaxe do código sem a necessidade de enviá-lo para a placa. Isso permite que você identifique erros de digitação, problemas de sintaxe ou qualquer outra anomalia no código antes de executá-lo.

Carregar: Uma vez que o código tenha sido verificado e não contenha erros sintáticos, a próxima etapa é o carregamento. Isso implica enviar o código compilado para o microcontrolador. Essa ação resulta na execução real do programa na placa, permitindo que você observe o comportamento do seu código em um ambiente real.

Depurar: A depuração é um processo muito importante para identificar e corrigir erros lógicos ou funcionais no código. Ao rastrear a execução do programa, você pode identificar onde e por que o código não está funcionando conforme o esperado. Isso envolve a identificação de valores de variáveis, inspeção de fluxo de controle e análise de saídas.

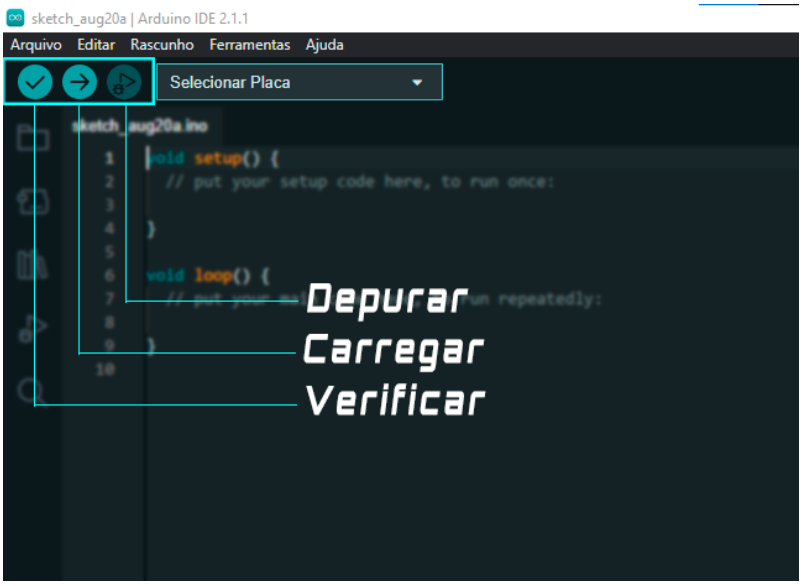

Figura 27: Ferramentas de execução e compilação do código

Essas três etapas, verificação, carregamento e depuração, compõem um ciclo essencial no desenvolvimento de software. Elas permitem que você refine e otimize seu código, garantindo que ele funcione de acordo com suas intenções. O uso dessas ferramentas é imperativo para garantir a confiabilidade e a eficácia do seu programa.

9.3 Estrutura de Programa

Com as ferramentas definidas, o próximo passo é desenvolver o sistema para a aplicação IoT. Geralmente, um programa para dispositivos IoT segue uma estrutura padrão, organizada em três partes principais:

1. Inicialização do hardware: Inclui a configuração de sensores, atuadores e módulos de conectividade, como Wi-Fi ou Bluetooth. Exemplo: Configurar um sensor DHT22 para medir temperatura e um módulo Wi-Fi para enviar os dados.

2. Rotina principal: Consiste em um loop que realiza tarefas repetitivas, como: realizar leitura de dados de sensores, processamento e filtragem de dados, comunicação com outros dispositivos ou servidores.

3. Interação com servidores/cloud: Envolve a transmissão de dados para uma plataforma em nuvem usando protocolos como MQTT ou HTTP, permitindo o monitoramento remoto.

Essa abordagem modular permite a criação de programas eficientes, escaláveis e fáceis de manter, mesmo em sistemas complexos com múltiplos dispositivos.

10 Protocolos de comunicação e segurança

Os protocolos de comunicação e segurança desempenham papéis centrais no envio e recepção de dados, garantindo que informações sejam transmitidas de maneira organizada, confiável e protegida contra acessos não autorizados. Enquanto os protocolos de comunicação definem como os dados são formatados, transmitidos e interpretados, os protocolos de segurança protegem essas interações contra ameaças externas, como interceptações e ataques cibernéticos. A combinação desses dois aspectos possibilita o funcionamento adequado de redes que interligam desde pequenos sensores até grandes infraestruturas.

Nesta seção, exploraremos os principais protocolos utilizados em sistemas IoT, detalhando suas funções, características e aplicações. Além disso, abordaremos os protocolos de segurança, destacando sua importância na proteção de dados e dispositivos conectados.

10.1 Protocolos de comunicação

Os protocolos de comunicação possibilitam a troca de informação entre dispositivos por meio da padronização de conjuntos de regras e convenções em uma rede. Estes apresentam extrema importância na atualidade, pois possibilitam uma transferência de dados de forma organizada, segura e sem erros. A função principal dos protocolos de comunicação é definir como os dados devem ser formatados, transmitidos, recebidos e processados, assegurando que todas as partes envolvidas compreendam e possam utilizar a informação trocada. Existe uma enorme diversidade de protocolos, cada um desenvolvido para atender às necessidades específicas de diferentes tecnologias e aplicações, desde redes locais até a internet como um todo.

Os sistemas de IoT envolvem comunicações entre dispositivos, muitas vezes com características e restrições distintas. Para organizar essa complexidade, os protocolos de comunicação são estruturados em níveis ou camadas, onde cada nível desempenha um papel específico no transporte e na interpretação dos dados. Esta abordagem modular permite que diferentes tecnologias colaborem para formar uma rede robusta, escalável e eficiente.

Embora existam várias abordagens para descrever os níveis de comunicação, o modelo OSI (Open Systems Interconnection) é uma referência amplamente utilizada. Ele divide o processo de comunicação em sete camadas:

- Física: Responsável pela transmissão física dos bits por meio de cabos, ondas de rádio e etc.
- Enlace: Responsável por organizar os dados em quadros (frames) e os erros físicos são detectados e corrigidos.

- Rede: Responsável por realizar a identificação dos dispositivos na rede e fornecer o caminho para o dado.
- Transporte: Responsável por garantir que os dados cheguem ao destino de forma confiável e na ordem correta.
- Sessão: Responsável pela troca de dados entre aplicações.
- Apresentação: Converte os dados recebidos pela camada de aplicação para um formato compatível com o seu transporte.
- Aplicação: Responsável por fazer a comunicação entre a máquina e o usuário.

No contexto de IoT, é usado normalmente o modelo TCP/IP que compila essas camadas em apenas quatro.

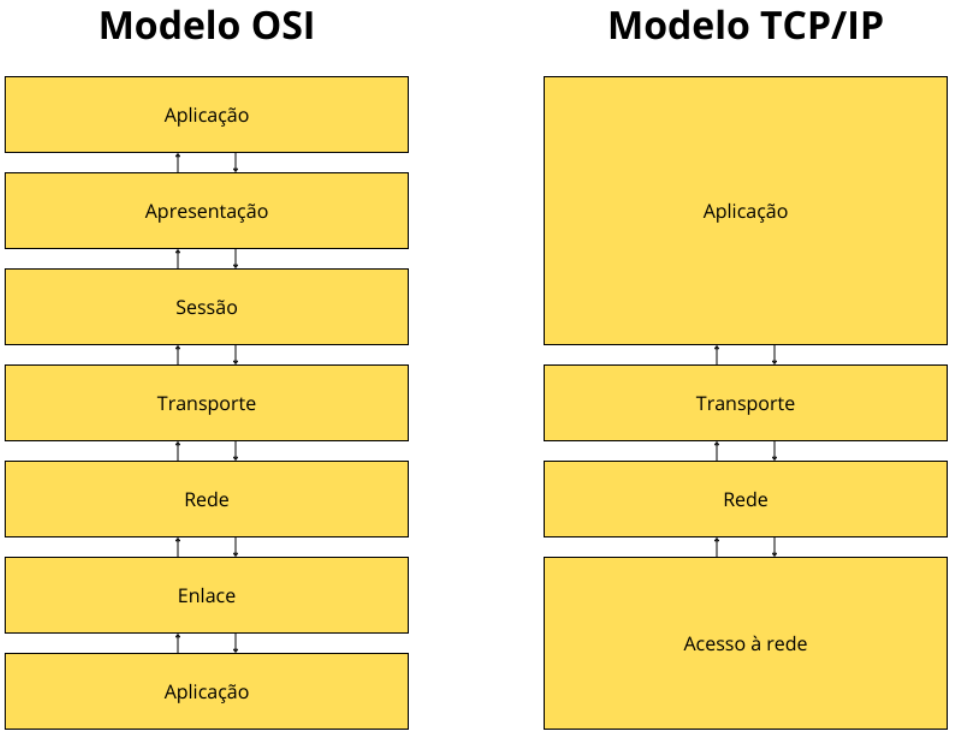

Figura 28: Comparação entre os modelos de redes

Abaixo serão apresentados os principais tipos de prótocolos usados em cada camada do modelo TCP/IP em aplicações IoT.

10.1.1 Camada de Acesso à Rede

Como visto acima, essa camada implementa tanto o nível físico quanto o nível de enlace. Em aplicações IoT, os protocolos dessa camada são escolhidos com base nas necessidades de alcance, consumo de energia e taxa de transmissão de dados. Os principais protocolos usados são:

- Ethernet: Utilizado em redes cabeadas, é confiável e oferece altas taxas de transmissão. É mais comum em aplicações IoT industriais.
- Wi-Fi (IEEE 802.11): Amplamente usado em ambientes domésticos e urbanos, permite altas velocidades de conexão, mas consome mais energia, sendo adequado para dispositivos conectados à rede elétrica.
- Bluetooth e BLE (Bluetooth Low Energy): Com alcance curto e baixo consumo de energia, é ideal para dispositivos portáteis, como wearables e sensores.
- LoRaWAN: Projetado para comunicações de longa distância com baixo consumo de energia, é utilizado em aplicações IoT rurais ou urbanas, como monitoramento ambiental e cidades inteligentes.
- Zigbee: Popular em automação residencial e industrial, oferece baixo consumo de energia e suporte para redes mesh.
- NB-IoT e LTE-M: Protocolos de comunicação celular especializados em IoT, permitindo conexão direta com a infraestrutura das operadoras de telecomunicação. São adequados para aplicações com baixa latência e alta confiabilidade.

10.1.2 Camada de Internet

A Camada de Internet é responsável por implementar o nível de rede. Em IoT, essa camada conecta dispositivos de redes locais (LAN) com a infraestrutura global da Internet, utilizando endereços lógicos como IPv4 ou IPv6. Os principais protocolos usados são:

- IPv4 (Internet Protocol versão 4): Amplamente utilizado, mas com limitação no número de endereços disponíveis, o que é um desafio para o crescente número de dispositivos IoT.
- IPv6: Projetado para superar as limitações do IPv4, oferece um espaço de endereçamento muito maior e recursos adicionais, como configuração automática e melhor suporte para multicast, sendo cada vez mais usado em aplicações IoT.

- RPL (Routing Protocol for Low-Power and Lossy Networks): Um protocolo de roteamento específico para redes de IoT com dispositivos de baixa potência, como sensores. Ele otimiza o uso de energia e suporta topologias como redes mesh.

A adoção de IPv6 tem sido fundamental em IoT devido ao grande número de dispositivos que precisam ser conectados diretamente à internet.

10.1.3 Camada de Transporte

A Camada de Transporte implementa o nível de mesmo nome do modelo OSI. Dessa forma, gerencia a confiabilidade da comunicação entre os dispositivos, controlando o envio e recebimento de dados. Ela também garante que as mensagens sejam entregues corretamente, mesmo que ocorram problemas na rede. Os principais protocolos usados são:

- TCP (Transmission Control Protocol): Protocolo confiável, que verifica erros, controla o fluxo e retransmite pacotes perdidos. Ideal para aplicações IoT onde a integridade dos dados é essencial, como monitoramento de saúde.
- UDP (User Datagram Protocol): Protocolo não confiável, mas mais rápido e leve. É usado em cenários onde a velocidade é mais importante que a precisão, como transmissões em tempo real de sensores ambientais.
- QUIC (Quick UDP Internet Connections): Um protocolo mais recente, que combina a velocidade do UDP com maior segurança e confiabilidade. Pode ser usado em sistemas IoT avançados.

A escolha entre TCP e UDP depende do caso de uso. Por exemplo, TCP é adequado para dispositivos que enviam dados críticos para a nuvem, enquanto UDP é preferido para aplicações em tempo real com baixo consumo de energia.

10.1.4 Camada de Aplicação

A Camada de Aplicação é onde a interação direta com os dados ocorre. Nesta camada, os protocolos definem como os dispositivos IoT trocam mensagens, acessam servidores e interpretam os dados enviados e recebidos. Os principais protocolos usados são:

- HTTP/HTTPS (Hypertext Transfer Protocol): Comumente usado em APIs e integração com sistemas legados. O HTTPS, que inclui criptografia TLS/SSL, é preferido em aplicações que exigem segurança.

- MQTT (Message Queuing Telemetry Transport): Um protocolo leve, otimizado para IoT, ideal para dispositivos com recursos limitados e conexões intermitentes. Usado amplamente em automação residencial e monitoramento remoto.
- CoAP (Constrained Application Protocol): Similar ao HTTP, mas projetado para dispositivos com restrições de recursos. Ele funciona sobre UDP e é usado em aplicações como redes de sensores.
- WebSockets: Protocolo que mantém uma conexão persistente para comunicação bidirecional em tempo real, adequado para aplicações como automação industrial.
- AMQP (Advanced Message Queuing Protocol): Utilizado em sistemas IoT complexos, permite comunicação confiável e escalável entre dispositivos e servidores.

O protocolo escolhido nesta camada afeta diretamente a eficiência e segurança da comunicação em aplicações IoT. Por exemplo, MQTT é altamente usado para dispositivos com conexões instáveis, enquanto HTTPS é preferido em aplicações que lidam com dados sensíveis.

10.2 Protocolos de segurança

Imagine uma casa inteligente capaz de gerenciar o ar-condicionado, as luzes, a televisão e diversos outros dispositivos conectados. Nesse cenário, a segurança é essencial para evitar que pessoas não autorizadas acessem e controlem esses dispositivos ou capturem informações sensíveis. Para isso, é indispensável o uso de protocolos de segurança robustos, que garantem a proteção das conexões e dos dados transmitidos entre os dispositivos IoT.

Nesta seção, serão apresentados os principais protocolos de segurança utilizados em sistemas IoT, como DTLS, TLS/SSL, IPsec, OAuth 2.0 e X.509 Certificates, detalhando suas funções e aplicações na proteção de redes e dispositivos conectados.

10.2.1 DTLS

DTLS é o acrônimo na língua inglesa para Datagram Transport Layer Security. Traduzindo seu significado, ele age nas camadas de transporte da rede que usam o protocolo de transporte UDP. Seu objetivo é proteger o transporte de datagramas, que são a forma com que as informações são transmitidas na internet. Como o próprio nome diz, ele é baseado no Transport Layer Security (TLS), que é um protocolo de criptografia cuja função é promover segurança para as comunicações

feitas por meio de uma rede de computadores.
O DTLS é adequado para proteger aplicações e serviços que são sensíveis a atrasos, aplicações de túnel como VPNs e aplicações que tendem a ficam sem descritores de arquivo ou buffers de soquete. Ele permite que aplicativos cliente e servidor se comuniquem de uma maneira concebida para combater espionagem, sabotagem ou falsificação de mensagem.
Seus benefícios incluem um baixo overhead (parte do pacote que não é aproveitada para transmitir dados) e uma latência reduzida. Por outro lado, por ser um protocolo de datagrama, não há garantia da ordem da entrega das informações, nem que elas sejam todas entregues.
Um exemplo de uso do DTLS é no Constrained Application Protocol (CoAP). O CoAP é feito para aplicações Machine to Machine (M2M) para uso em nós e redes de capacidade de hardware e potências baixas. Exemplos de dispositivos de potência e processamento limitados são os microcontroladores e Single Board Computers (SBCs), utilizados para a observação de sensores e controle de atuadores, e conectados à internet através de dispositivos de gateway devido a sua limitação de processamento.

10.2.2 TLS/SSL

TLS e SSL são dois certificados que aplicam protocolos de segurança baseados em criptografia de informações em sites. O SSL foi o primeiro certificado a usar esse tipo de protocolo de segurança, revolucionando os parâmetros de proteção no acesso a sites. No entanto, em determinado momento a sua capacidade de cumprir com a criptografia de informações foi testada e houve o entendimento de que era necessário desenvolver a tecnologia. Foi assim que o TLS passou a ser o certificado principal, praticamente substituindo o SSL, já que ele é um recurso mais avançado e mais seguro.
Transport Layer Security, o TLS, é um protocolo de segurança aplicado a web sites para garantir que todas as atividades realizadas naquele ambiente estejam protegidas contra ataques e vazamentos. Ele não impede consultas de domínios feitas pelo WHOIS, mas protege dados financeiros e pessoais de usuários que acessam e-commerces, sites e blogs.
A base do funcionamento do TLS como um protocolo de segurança é sua capacidade de codificar mensagens e informações. Assim, quando um usuário navega por um site e informa dados como endereço, número de telefone, código de segurança do cartão de crédito, entre outros, o TLS é capaz de “esconder” esses registros de quem não deveria ter acesso a eles.
O protocolo faz isso codificando qualquer mensagem que vai de um ponto a outro, ou seja, do site ao usuário e vice-versa. São criadas chaves de acesso às quais só emissor e receptor têm acesso e, quando uma informação faz esse caminho, o certificado TLS

autentica quem tentou acessar o conteúdo. Caso não seja um desses dois pontos, não há a possibilidade de acessar a mensagem.
As mensagens são criptografadas, ou seja, seu conteúdo original é escondido para que, por exemplo, em um ataque hacker, a informação interceptada não possa ser lida. Isso garante que todas as atividades realizadas em um site com o certificado TLS se mantenham protegidas, acessíveis apenas por emissor e receptor.

10.2.3 OAuth 2.0

OAuth significa Open Autorization, e é um padrão criado para permitir que um site ou aplicação acesse recursos hospedados por outros aplicatios de web, em nome do usuário. OAuth 2.0 fornece acesso consentido e restringe ações que o aplicativo cliente pode realizar nos recursos em nome do usuário, sem nunca compartilhar as credenciais do usuário.(Citar site do Auth0)
Apesar de a web ser a principal plataforma para o OAuth 2, outros tipos
O OAuth 2.0 é um protocolo de autorização, e não um protocolo de autenticação. Isso significa que ele foi concebido para conceder acesso a um conjunto de recursos, como APIs remotas ou dados de utilizadores.
OAuth 2.0 usa Tokens de Acesso, ou seja, pedaços de informação que representam a autorização para acessar recursos no nome do usuário final. O OAuth 2.0 não define um formato específico para os Tokens de Acesso, sendo o JSON Web Token (JWT) o mais frequente.
Dentro do OAuth 2.0, existem 4 papeis diferentes: Proprietário do Recurso (Resource Owner), Cliente (Client), Servidor de Autorização (Authorization Server) e Servidor de recursos (Resource Server). O primeiro é o usuário ou sistema que possui os recursos protegidos e pode conceder acesso a eles; o segundo é o sistema que requer acesso aos recursos protegidos, que somente serão acessados se o cliente possuir o Token de Acesso; o terceiro recebe pedidos por Tokens de Acesso e os emite após a autenticação bem sucedida e o consentimento do Proprietário do recurso; por fim, o quarto protege os recursos do usuário e recebe solicitações de acesso do cliente, aceitando e validando um Token de Acesso do Cliente e retorna os recursos apropriados a ele.

10.2.4 X.509 Certificates

X.509 é um formato padrão para certificados de chave pública, documentos digitais que associam com segurança pares de chaves criptográficas a identidades como sites, indivíduos ou organizações. Os aplicativos comuns dos certificados X.509 incluem:

- SSL /TLS e HTTPS para navegação na web autenticada e criptografada

- E-mail assinado e criptografado via S/MIME protocolo
- Assinatura de código
- Assinatura de documento
- Autenticação de cliente
- ID eletrônico emitido pelo governo

Independentemente das aplicações pretendidas, cada certificado X.509 inclui um chave pública, assinatura digital e informações sobre a identidade associada ao certificado e sua emissão pela autoridade de certificação (CA).
O chave pública faz parte de um par de chaves que também inclui um chave privada. A chave privada é mantida segura e a chave pública é incluída no certificado. Este par de chaves pública/privada:

- Permite que o proprietário da chave privada assine documentos digitalmente; essas assinaturas podem ser verificadas por qualquer pessoa com a chave pública correspondente.
- Permite que terceiros enviem mensagens criptografadas com a chave pública que somente o proprietário da chave privada pode descriptografar.

A assinatura digital é um hash codificado (resumo de comprimento fixo) de um documento que foi criptografado com uma chave privada. Quando um certificado X.509 é assinado por um CA publicamente confiável, como SSL.com, o certificado pode ser usado por terceiros para verificar a identidade da entidade que o apresenta. Cada certificado X.509 inclui campos que especificam o sujeito, CA de emissão, e outras informações necessárias, como o certificado versão e período de validade. Além disso, os certificados v3 contêm um conjunto de extensões que definem propriedades como usos de chave aceitáveis e identidades adicionais para vincular um par de chaves.

11 Propostas de Projeto

Agora iremos sintetizar tudo que foi abordado na apostila com um projeto prático onde vocês terão acesso a um passo a passo para conhecer novas ferramentas e entender como operar em problemas e desafios reais.

11.1 Ideia de projeto: Plantinha Inteligente

Neste projeto, guiaremos você passo a passo na criação de um sistema que monitora o nível de umidade do solo de uma planta. O sistema enviará os dados coletados para uma plataforma que os armazenará na nuvem, permitindo o controle de dispositivos conectados de forma remota. Além disso, implementaremos o controle de um LED, que poderá ser ligado e desligado remotamente, possibilitando o ajuste da luminosidade para o cuidado com a plantinha. Também será possível criar dashboards para monitoramento em tempo real e gerenciar automações de maneira prática, seja por meio de aplicativos móveis ou interfaces web. Dessa forma, você integrará tecnologia e praticidade em uma solução personalizada para a saúde e o bem-estar das suas plantas.

11.1.1 Materiais Físicos:

- ESP32 ou ESP8266;
- LEDs (para imitar uma lâmpada);
- Resistor 220 ohm;
- Sensor de umidade do solo;
- Protoboard;
- Jumpers;

11.1.2 Softwares e Interfaces:

- Blynk;
- Arduino IDE;

11.1.3 Linguagens:

- C++;

Parte 1 - Arquitetura do Projeto

Neste capítulo, apresentaremos a arquitetura de um sistema IoT projetado para monitorar e controlar o nível de umidade do solo de uma planta. O fluxo do sistema inicia-se com o sensor de umidade do solo, que mede os níveis de umidade e transmite os dados ao controlador ESP32 ou ESP8266. Esse microcontrolador processa as informações recebidas e as envia para a plataforma Blynk, que analisa os dados e os exibe graficamente em sua interface de usuário. Essa abordagem proporciona um monitoramento visual claro, eficiente e acessível.

Além do monitoramento, o sistema oferece controle remoto integrado. Um LED, que atua como indicador luminoso para ajustar a luminosidade da planta, pode ser acionado diretamente pela interface do Blynk. Essa funcionalidade permite uma resposta rápida e intuitiva, evidenciando a interação entre os elementos do sistema.

Essa arquitetura demonstra a integração eficiente entre sensores, controlador, plataforma de IoT e interface de usuário, resultando em um sistema robusto, funcional e prático para aplicações de monitoramento e controle remoto.

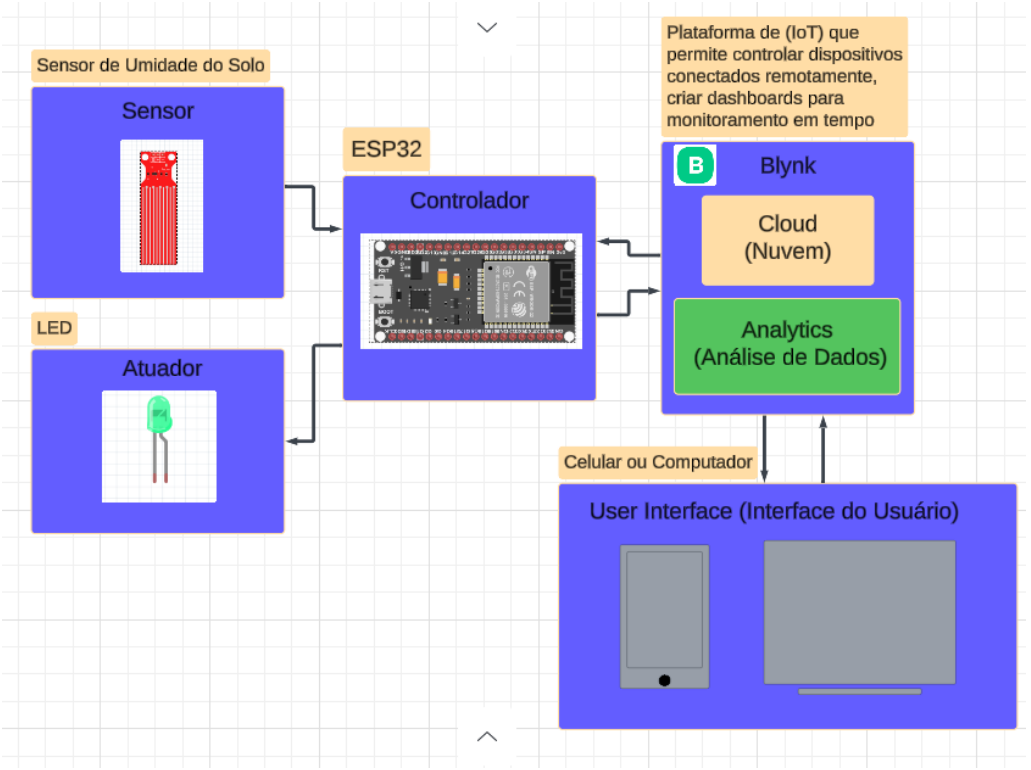

Figura 29: Arquitetura do projeto

Para realizar essa configuração é essencial entender o que cada pino do seu microcontrolador faz. Para isso, é fundamental compreender a pinagem do ESP32 e do ESP8266, os dois microcontroladores amplamente utilizados em projetos de IoT. Abaixo, apresentamos os esquemas de pinagem de ambos, destacando suas principais características.

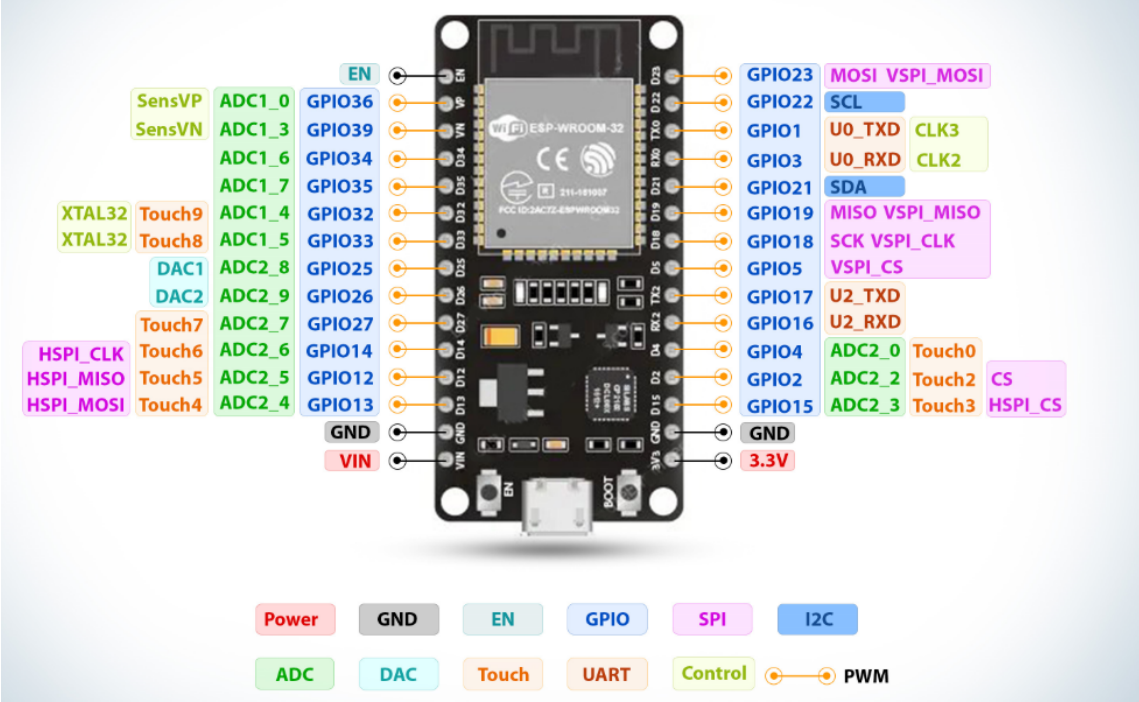

Figura 30: Pinagem do ESP32

O **ESP32** é um microcontrolador poderoso, com múltiplos pinos GPIO configuráveis como entradas ou saídas digitais. Ele conta com ADCs para leitura de sinais analógicos, suporte para comunicação via I2C, SPI, UART e até DACs para gerar sinais analógicos. É amplamente utilizado em projetos que demandam conectividade Wi-Fi e Bluetooth, com uma arquitetura que permite maior flexibilidade para controle e monitoramento de dispositivos.

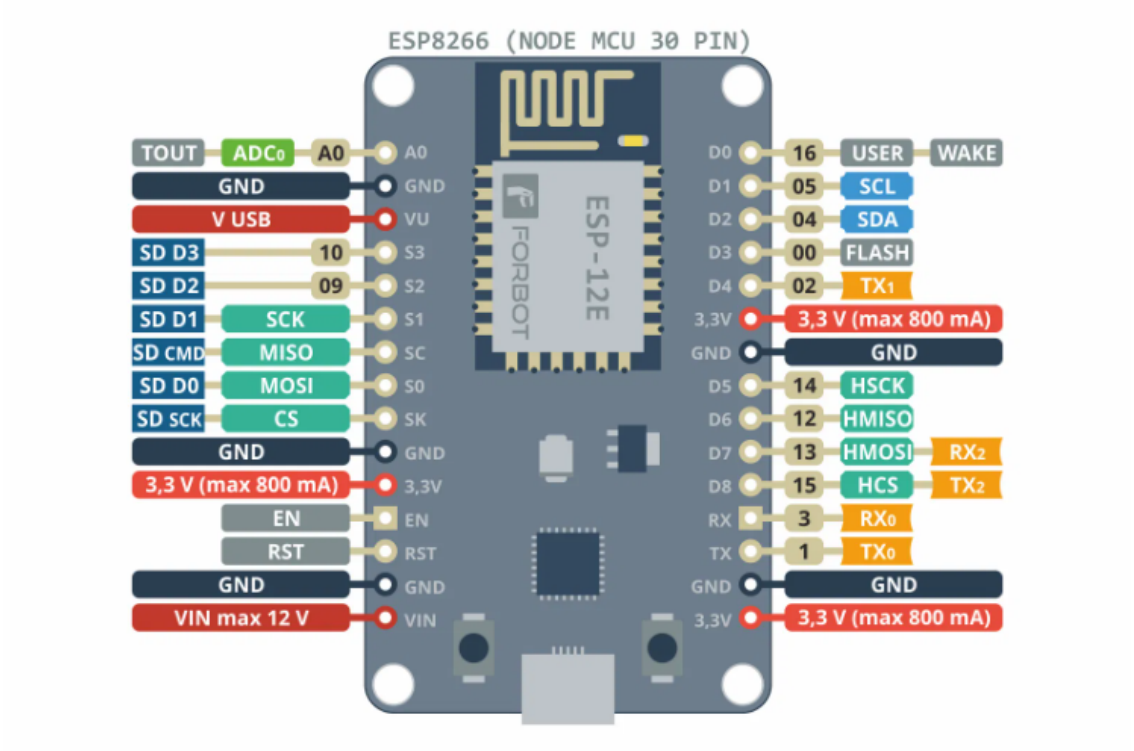

Figura 31: Pinagem do ESP8266

O **ESP8266**, por sua vez, é uma solução compacta e eficiente para projetos de IoT. Apesar de contar com menos GPIOs que o ESP32, eles são ideais para aplicações que exigem conectividade Wi-Fi em um formato simples e econômico. Seus pinos permitem comunicação serial, leitura de sensores analógicos e controle de dispositivos digitais.

Para entender o significado de cada um dos rótulos, preparamos uma tabela que descreve detalhadamente a função de cada pino. Com essas pinagens em mente, você será capaz de planejar as conexões necessárias para o seu projeto, garantindo que cada componente esteja corretamente conectado ao microcontrolador escolhido:

Tabela 1: Significado do Pinout de um Microcontrolador

Nome do Pino	**Descrição**
GPIO	Pinos de entrada/saída digitais configuráveis para leitura de sensores ou controle de atuadores.
VCC	Fornece a tensão de alimentação ao microcontrolador (geralmente 3.3V ou 5V).
GND	Conexão de terra comum para o funcionamento dos circuitos.
ADC	Pino para conversão de sinais analógicos em digitais, utilizado em sensores analógicos.
DAC	Converte sinais digitais em saídas analógicas, útil em LEDs RGB e outros dispositivos.
PWM	Saída para geração de sinais de largura de pulso, utilizado em controle de motores e LEDs.
UART	Comunicação serial assíncrona para envio/recebimento de dados (ex.: PC ou módulos).
I2C	Pinos para comunicação com dispositivos usando o protocolo I2C.
SPI	Interface de alta velocidade para comunicação com periféricos (ex.: displays, sensores).
EN	Ativa o microcontrolador, geralmente conectado à tensão alta para operação.
RST	Pino para reiniciar o microcontrolador.
Vin	Entrada de tensão externa para alimentar o microcontrolador.
BOOT	Pino para selecionar o modo de programação ou execução do microcontrolador.
TX	Pino de transmissão de dados em comunicação serial (comunicação com outros dispositivos).
RX	Pino de recepção de dados em comunicação serial.
SDA	Linha de dados do protocolo I2C.
SCL	Linha de clock do protocolo I2C.

Parte 2 - Configurando o Blynk

Agora vamos configurar o Blynk para se comunicar com o microcontrolador. A seguir, apresentamos um passo a passo detalhado:

1º Passo - Criar uma Conta na Plataforma Blynk

Acesse o site oficial da plataforma Blynk: `https://blynk.io/`

Crie uma conta com seu e-mail e senha. Após isso, faça login para acessar o painel principal.

2º Passo - Criar um Novo Projeto

Após entrar na plataforma, você verá a interface principal. Clique no botão verde chamado **"+ Novo Modelo"** para criar um novo projeto:

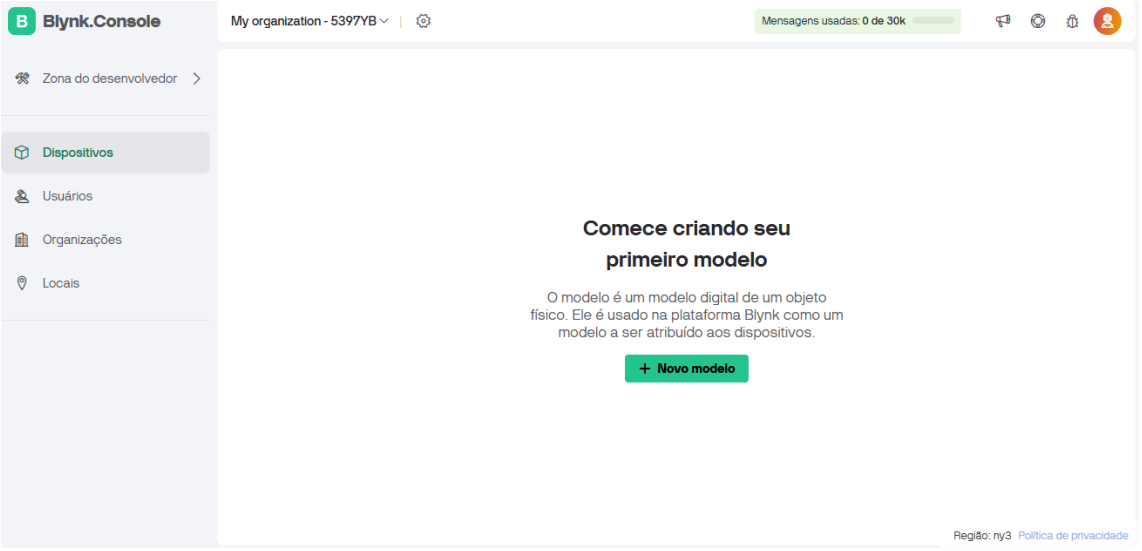

Figura 32: Tela inicial do Blynk.

Na tela de configuração inicial:

- **Nome do Projeto:** Defina um nome para o seu projeto.
- **Tipo de Microcontrolador:** Escolha o microcontrolador que você está utilizando (ESP32, ESP8266, etc.).

- **Tipo de Conexão:** Selecione o método de conexão (Wi-Fi, Ethernet, etc.).

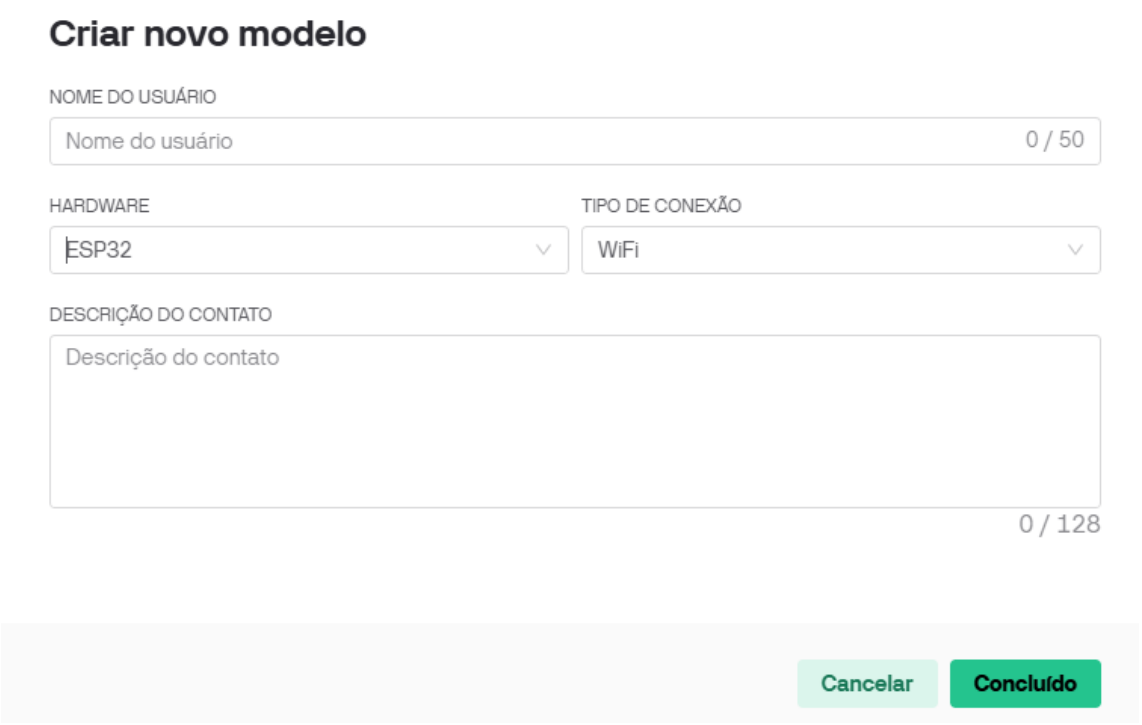

Figura 33: Configurações iniciais do projeto.

3º Passo - Configurar os Fluxos de Dados (DataStreams)

Agora é necessário configurar os fluxos de dados (DataStreams) para o projeto.

1. Clique em **"DataStreams"** no menu lateral.
2. Selecione **"Novo DataStream"** e escolha a opção **"Pino Virtual"**.
3. Configure os DataStreams para cada componente (sensores ou atuadores) que deseja monitorar ou controlar.

4º Passo - Criar um Novo Dispositivo

Neste passo, criaremos as configurações de comunicação para o dispositivo físico.

Pino	Pino Virtual	Nome de Usuário	Faixa de Valores	Valor Padrão
P1	V0	Sensor de Umidade	0 a 100	0
P2	V1	LED	0 a 1	0
P3	V2	Botão	0 a 1	0

Figura 34: Configuração de DataStreams no Blynk.

1. Acesse a aba ou guia **"Dispositivos"**.
2. Clique em **"Novo Dispositivo"**.
3. Escolha a opção **"A partir de um modelo"** e selecione o modelo configurado anteriormente.

Ao concluir, será gerado um código de autenticação. Salve esse código, pois será necessário para o código no microcontrolador.

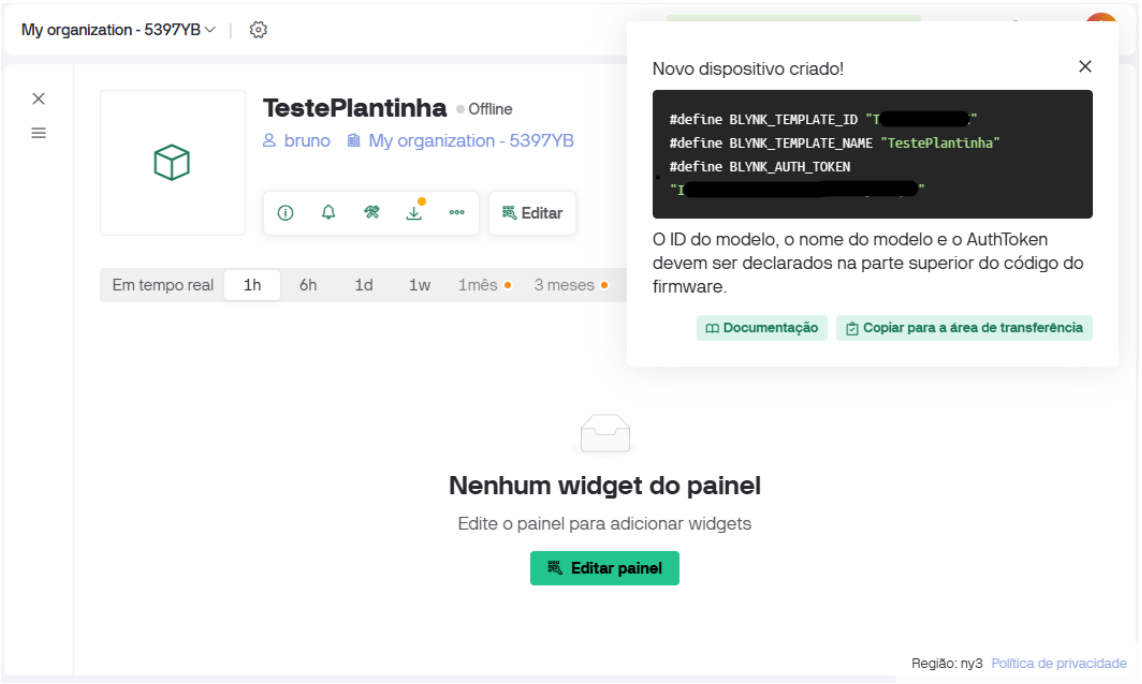

Figura 35: Criação de um novo dispositivo.

5º Passo - Editar o Painel de Controle

Agora, vamos configurar o painel de controle para visualização e interação com os dispositivos:

1. Clique na aba **"Editar Painel"**.
2. Adicione os componentes desejados arrastando-os para o painel. Exemplos:
   - **LED:** Indicador luminoso.
   - **Medidor:** Para exibir os dados do sensor de umidade.
   - **Interruptor:** Para controlar dispositivos remotamente.
3. Configure cada componente clicando no ícone de engrenagem e associando-o ao DataStream correspondente:
   - LED → DataStream do LED.
   - Medidor → DataStream do sensor de umidade.
   - Interruptor → DataStream do botão.

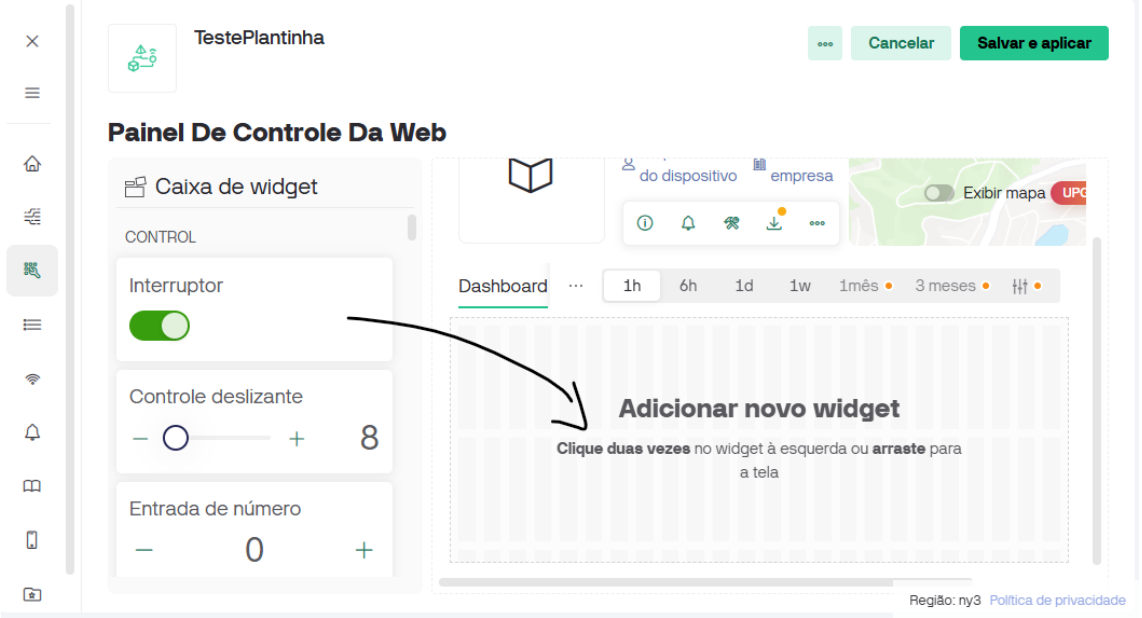

Figura 36: Configuração do painel de controle.

Finalizando a Configuração do Blynk

Com isso, a configuração do Blynk está concluída. No próximo passo, utilizaremos o código gerado para conectar o microcontrolador ao projeto e começar a interagir com o sistema.

Parte 3 - Código

Nesta seção, vamos configurar o ambiente de desenvolvimento e preparar o código necessário para a comunicação do seu microcontrolador com o Blynk. Siga as etapas abaixo para configurar e carregar o código corretamente.

1º Passo - Baixar e Instalar o Arduino IDE

A primeira etapa é instalar o Arduino IDE em seu computador. Na página inicial do Arduino IDE, você encontrará opções de download para diferentes sistemas operacionais. Selecione a opção correspondente ao seu sistema:

Figura 37: Página inicial do Arduino IDE.

2º Passo - Configurar a Plataforma para seu Microcontrolador

Caso seu microcontrolador ainda não esteja configurado na IDE do Arduino, siga os seguintes passos para adicionar a plataforma:

1. Vá para **Arquivo → Preferências**.
2. Na seção "URLs do gerenciador de placas adicionais", adicione a URL de download do pacote de bibliotecas correspondente ao seu microcontrolador:
   - Para ESP8266: `http://arduino.esp8266.com/stable/package_esp8266com_index.json`
   - Para ESP32: `https://dl.espressif.com/dl/package_esp32_index.json`
3. Clique em **OK**.

Após isso, você poderá selecionar a placa ESP8266 ou ESP32 em **Ferramentas → Placa**.

3º Passo - Baixar e Instalar as Bibliotecas para Comunicação com o Blynk

Agora, vamos instalar a biblioteca Blynk para que possamos utilizar a plataforma Blynk com nosso microcontrolador. Siga os passos abaixo:

1. Abra o Arduino IDE.
2. Vá até o menu **Sketch → Incluir Biblioteca → Gerenciar Bibliotecas**.
3. Na barra de pesquisa, procure por **Blynk**.
4. Clique em **Instalar** na biblioteca oficial do Blynk.

Com a biblioteca instalada, você poderá utilizá-la para comunicar o microcontrolador com a plataforma Blynk.

4º Passo - Preparar o Código de Comunicação IoT

A seguir, fornecemos um exemplo de código que você deve utilizar para conectar seu microcontrolador ao projeto Blynk. Apenas substitua alguns valores e faça as configurações necessárias:

1. **Substitua as variáveis de configuração Blynk:**

   Os dados de `BLYNK_TEMPLATE_ID`, `BLYNK_TEMPLATE_NAME` e `BLYNK_AUTH_TOKEN` devem ser substituídos pelos valores gerados pela plataforma Blynk. Lembre-se de que você já salvou esse código anteriormente, então é hora de utilizá-lo!

2. **Descomente as bibliotecas de acordo com o seu microcontrolador:** No código há dois pares de bibliotecas que irão gerenciar sua conexão WiFi e comunicação com o blynk, você deve utilizar o primeiro se estiver trabalhando com uma ESP8266 ou o segundo se estiver trabalhando com ESP32. Se você estiver usando um microcontrolador diferente, pesquise pela biblioteca correspondente e desconsidere as bibliotecas atuais.

3. **Configure o Wi-Fi:**

   No código, altere o `ssid` e a `senha` para os dados da sua rede Wi-Fi.

4. **Verifique a Pinagem Virtual:**

   As pinagens virtuais `V0`, `V1` e `V2` devem corresponder ao sensor, ao botão e ao LED, respectivamente. Assegure-se de que essas pinagens estejam configuradas corretamente no código.

5. **Carregue o código no microcontrolador:**

   Conforme ensinado anteriormente, selecione sua placa no Arduino IDE e carregue o código. Após isso, você poderá visualizar os dados na interface web do Blynk e controlar o LED da sua plantinha diretamente pela interface.

Assim, o sistema estará configurado para monitorar o sensor de umidade do solo e controlar o LED da planta remotamente via Blynk.

Listing 1: Título da Caixinha

```
#define BLYNK_PRINT Serial  // Enables Serial Monitor

#define BLYNK_TEMPLATE_ID "APMMSE3DoID34eD"
#define BLYNK_TEMPLATE_NAME "TestePlantinha"
#define BLYNK_AUTH_TOKEN "BDBAEWCFbmGBGBDFHXZCC54DS7Dy1G8qzF"
```

```
//descomente apenas as bibliotecas que for usar:

//#include <ESP8266WiFi.h>        //Para esp8266
//#include <BlynkSimpleEsp8266.h> //Para esp8266

//#include <WiFi.h>             //Para esp32
//#include <BlynkSimpleEsp32.h> //Para esp32


const char* ssid = "Nome da rede sua rede WiFi";
const char* pass = "Senha da sua rede WiFi";

#define SIGNAL_PIN A0   // Pino anal gico do sensor de  gua
#define LED_PIN 4       // Pino digital do LED
#define BOT 5
int valorAnterior = 0;


BLYNK_WRITE(V0) {
  Serial.begin(9600);
  digitalWrite(BOT, param.asInt());
  Serial.println(digitalRead(BOT));
}

void setup() {
  Serial.begin(9600);

  pinMode(BOT, OUTPUT);
  pinMode(A0, INPUT);
  pinMode(LED_PIN, OUTPUT);

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass, "blynk.cloud", 80);
}

void loop() {
  AscendeLED();
  VerificaSensor();
  Blynk.run();
  delay(500);
```

```
}

void AscendeLED() {
  int botLed = digitalRead(BOT);
  if (botLed) {
    digitalWrite(LED_PIN, HIGH);
    Serial.println("aceso");
    Blynk.virtualWrite(V2, HIGH);  // Atualiza o estado do LED virtual no
  } else {
    digitalWrite(LED_PIN, LOW);
    Blynk.virtualWrite(V2, LOW);  // Atualiza o estado do LED virtual no
  }
}

void VerificaSensor() {
  int value = analogRead(SIGNAL_PIN);  // L   o valor do sensor de  gua
  int value_map = map(value, 0, 1023, 0, 100);

  if (value_map != valorAnterior) {
    Blynk.virtualWrite(V1, value_map);  // Envia o valor do sensor para o
    valorAnterior = value_map;          // Atualiza o valor anterior
  }
}
```

Parte 4 - Conclusão

Isso é tudo, pessoal! Esperamos que tenham aproveitado o conteúdo desta apostila sobre IoT e que ela tenha despertado sua curiosidade e entusiasmo por esse vasto universo tecnológico. Este é apenas o início de uma jornada repleta de possibilidades, onde criatividade e conhecimento técnico se unem para solucionar desafios do mundo real.

Com os conceitos e ferramentas apresentados, você já possui uma base sólida para desenvolver sistemas inovadores e explorar o potencial da Internet das Coisas. Agora, o próximo passo é colocar a mão na massa: experimente alterar os códigos fornecidos, modificar as lógicas existentes e até mesmo criar seus próprios projetos. A prática é essencial para consolidar o aprendizado e impulsionar sua evolução.

Continue explorando, testando ideias e compartilhando suas descobertas. Afinal, o futuro da tecnologia é construído por mentes curiosas, motivadas e dispostas a

transformar conceitos em realidade. Boa sorte em sua jornada, e sucesso em seus projetos! Até a próxima!