Introdução à Internet das Coisas

Integração entre Dispositivos

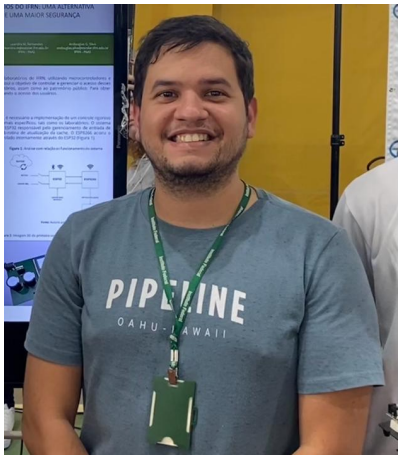
Apostila do aluno

Ph.D. Andouglas Gonçalves da Silva Júnior
Ph.D. Manoel do Bonfim Lins de Aquino
Marcos Fábio Carneiro e Silva
Autor da apostila

Ph.D. Andouglas Gonçalves da Silva Júnior
Ph.D. Manoel do Bonfim Lins de Aquino
Instrutor do curso

Larissa Jéssica Alves – Analista de Suporte Pedagógico
Revisão da apostila

Autor

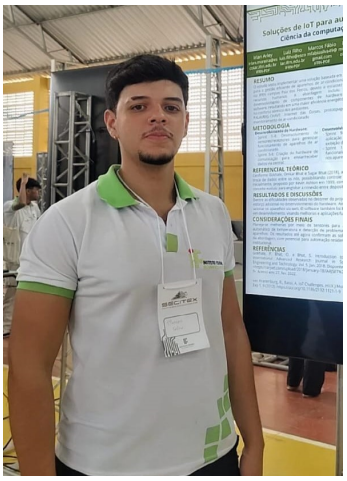


Andouglas Gonçalves da Silva Júnior

Doutor em Engenharia Elétrica e da Computação - UFRN. Mestre em Engenharia Mecatrônica na área de Sistemas Mecatrônicos. Bacharel em Ciências e Tecnologia pela EC&T - Escola de Ciências e Tecnologia - UFRN. Engenheiro Mecatrônico - UFRN. Professor de Ensino Básico, Técnico e Tecnológico no Instituto Federal de Educação Tecnológica do Rio Grande do Norte (IFRN). Integrante da Rede de Laboratórios NatalNet, LAICA e colaborador ISASI-CNR-Itália. Desenvolve projetos na área de Machine Learning, Internet das Coisas e Holografia Digital. Colabora no projeto do N-Boat (Veleiro Robótico Autônomo), principalmente no desenvolvimento de sistemas para monitoramento da qualidade da água e identificação de micropartículas usando holografia digital e IA.

Manoel do Bonfim Lins de Aquino

Possui graduação (2006), mestrado (2008) e doutorado (2022) em Engenharia Elétrica e da Computação, pela Universidade Federal do Rio Grande do Norte - UFRN. Tem experiência na área de projetos de Telecomunicações, atuando na Siemens como engenheiro de Telecomunicações (2008-2010). Sou Professor (2010 - atual) do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte - IFRN, membro do NADIC, Núcleo de Análise de Dados e Inteligência Computacional, onde venho desenvolvendo projetos de P&D nas áreas de desenvolvimento de sistemas, IoT e Inteligência Artificial.



Marcos Fábio Carneiro e Silva

Estudante de informática no Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte. Participou, como bolsista, de projeto de pesquisa voltado à automação em ambiente escolar com IoT (2022 - 2023), Possui experiência em redes de computadores, eletrônica e IoT, com ênfase na utilização de microcontroladores e desenvolvimento de Hardware. Além disso, possui conhecimentos básicos em Python e C++. Atualmente é membro do NADIC (Núcleo de Análise de Dados e Inteligência Computacional) do IFRN.

APRESENTAÇÃO

Bem-vindo ao curso de **Introdução à Internet das Coisas** do *CEPEDI!*

A Internet das Coisas, ou IoT, é uma revolução tecnológica que está transformando a maneira como interagimos com o mundo ao nosso redor. Essa inovadora e crescente rede de dispositivos interconectados, que variam desde sensores e aparelhos domésticos até veículos e equipamentos industriais, está desencadeando uma mudança fundamental em como coletamos, compartilhamos e utilizamos informações.

Neste curso, introduziremos o fascinante mundo da IoT, definindo conceitos básicos, seus principais componentes e aplicações. Além disso, introduziremos os protocolos de comunicação mais usados em aplicações de Internet das Coisas, além dos principais dispositivos utilizados hoje, como ESP32, Arduino, Raspberry Pi Pico, entre outros.

Além disso, pretendemos oferecer um curso que mescle a teoria com a prática. Para isso, utilizaremos aplicações livres como Wokwi e Bipes para desenvolvimento de projetos que nos auxiliarão no aprendizado dos conceitos teóricos.

Recomendamos ao aluno que, ao final da leitura de cada seção, realize os exercícios propostos e acesse os materiais indicados nas referências bibliográficas, para aprofundar a leitura desse material e complementar o que foi lido aqui.

Desejo a você, prezado aluno, que tenha um excelente curso!!

Boa Leitura !!

Sumário

1 Integração de dispositivos e sistemas em soluções de IoT

Agora que já conhecemos um pouco mais do mundo da Internet das Coisas, podemos aprofundar um pouco mais e tentar entender como é realizada a comunicação entre esses dispositivos a nível de hardware. Lembre-se que nesta fase, quando falamos de comunicação, não estamos nos referindo a uma conversas entres dispositivos IoT e sim entre componentes do mesmo dispositivo. Por exemplo, se temos um microncontrolador que recebe dados de um módulo sensor de temperatura, neste capítulo, aprenderemos como essa comunicação “interna” acontece. Mais pra frente veremos a camada de rede de um solução IoT e então discutiremos as diferentes formas de comunicar dispositiovos IoT.

1.1 Comunicação de dados entre Dispositivos

Vimos anteriormente que a arquitetura de uma aplicação de internet das coisas consiste em diferentes dispositivos interligados entre si, trocando informações (dados) por meio de uma rede. Além disso, também é importante entender como dispositivos que fazem parte de uma mesma estrutura (invólucro) se comunicam entre si. Por exemplo, vamos supor que queremos criar uma aplicação em IoT que monitore a temperatura de diferentes ambientes em uma empresa de produção de lacticínios. Poderíamos pensar na nossa arquitetura com vários dispositivos que medissem a temperatura, localizados nos diferentes ambientes da empresa, e que enviassem os dados por meio de uma rede a uma central de controle de dados.

Podemos usar um módulo sensor de temperatura ligado a um ESP 32 para ser esse dispositivo de medição. Vimos que os sensores são capazes de “perceber” as mudanças de uma variável e transformá-las em um sinal elétrico. Esse sinal seria lido pelo microcontrolador, que enviaria os dados para a

central. Como faremos para receber a informação do módulo de temperatura? Será que esse módulo envia sinal digital ou analógico, ou esses dados são enviados embutidos em uma mensagem que precisa ser recebida e interpretada pelo microcontrolador?

Para entender como é feita esse envio de mensagens, vamos primeiro entender o que é um sinal, seus tipos e como eles podem ser usados.

1.2 Sinal Elétricos

Nos referimos a um sinal elétrico como uma "variação mensurável de uma grandeza elétrica ao longo do tempo". Em outras palavras, um sinal é uma função de uma ou mais variáveis, a qual se veicula informações através de energia eletromagnética entre o transmissor e receptor. Pode representar informações, como dados ou mensagens, e é fundamental em sistemas elétricos e eletrônicos para transmitir informações e controlar dispositivos.

Esse sinais podem ser transportados por diferentes meios, como fios condutores ou através do ar via ondas eletromagnéticas e dependendo das características de como o dado será transmitido em função do tempo, podem ser do tipo **digital** ou **analógico**.

1.2.1 Sinal Digital

Um sinal digital é caracterizado pela fato de sua amplitude se manter sempre constante durante um intervalo de tempo. Na transmissão de sinais digitais é utilizado intervalos de tempo com a mesma duração (intervalo de sinalização) , onde em cada intervalo o sinal pode ter um valor diferente, mas que é fixo dentro do intervalo.

Um exemplo de um sinal digital bastante comum é quando este pode ter dois possíveis valores, 0 ou 1, como pode ser visto na Figura 3.1. Perceba que o eixo X corresponde ao tempo, e cada valor (0 ou 1) está contido dentro de um intervalo de tempo, como a definição dada anteriormente.

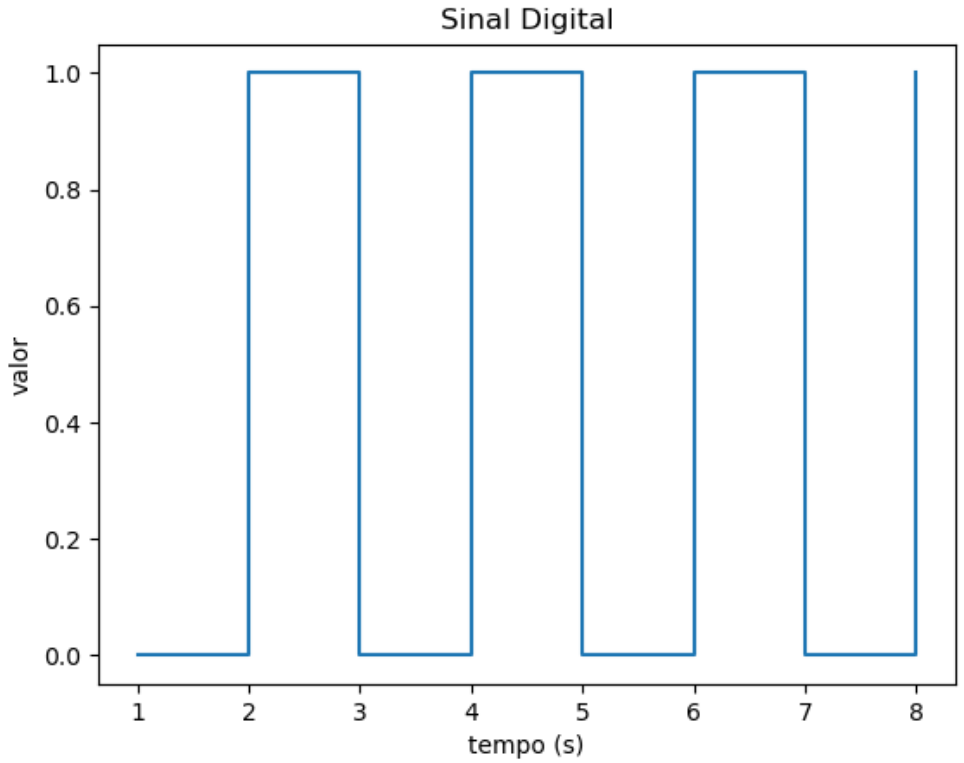


Fig. 3.1 - Representação de um sinal digital. (Autoria Própria)

Sinais digitais são muito comuns quando usamos microcontroladores. Como vimos nos capítulos anteriores, podemos utilizar pinos digitais para perceber a variação de um sinal contínuo. Por exemplo, um sensor de fim de curso pode ser ligado a uma entrada digital de forma que quando acionado, emita um sinal de nível alto para o microcontrolador, indicando que o sistema alcançou o final do curso.

1.2.2 Sinal Analógico

Quando falamos de um sinal analógico estamos nos referindo a amplitude um sinal que varia continuamente ao longo do tempo. Em outras palavras, existe um valor diferente para cada instante de tempo.

Sinais analógico são muito comuns, visto que muitas das grandezas do mundo real possui características analógicas. Por exemplo, a temperatura de um ambiente não muda de 20 ºC a 30 ºC de uma hora para outra. Essa temperatura aumenta, gradativamente, a medida que o ambiente esquenta. Ou

ainda, o nível de um reservatório não muda de uma coluna de água de 3m para 50 cm instantaneamente, ela depende da vazão constante da saída de água.

A Figura 3.2 mostra um sinal analógico. Percebe que para todos os instantes de tempo ao longo do eixo x existe um valor correspondente no eixo y que muda de forma contínua.

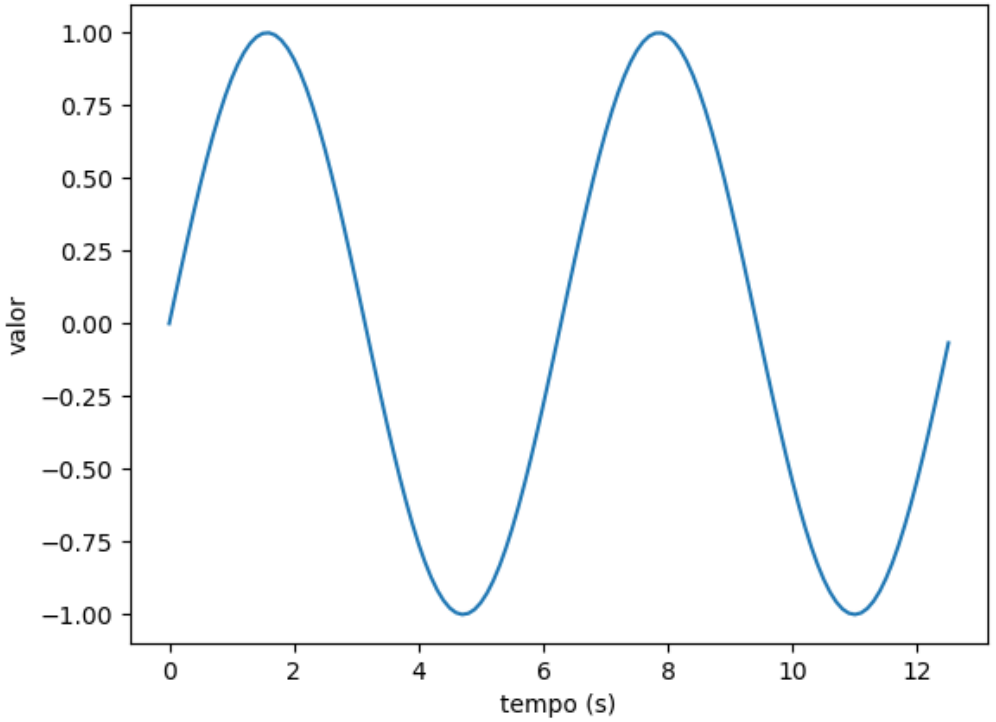


Fig. 3.2 - Representação de um sinal analógico. (Autoria Própria)

Um ponto importante é que, na verdade, as portas do microcontrolador só reconhecem sinais digitais, mesmo quando uma entrada fornece um sinal analógico. Nesses casos, o pino possui um conversor analógico-digital (ADC) que converte o sinal de entrada analógico em um sinal digital que pode ser “entendido” pelo controlador.

Por exemplo, quando um sinal de um LDR (sensor de luminosidade resistivo) é ligado a um pino analógico de 10 bits, os valores de resistência são convertidos para um sinal digital com 1024 níveis lógicos (2 elevado a 10). Se considerarmos um sinal que varia de 0 a 5V ligao ao pino de entrada de um microcontrolador, teremos 5 dividido por 1024, que resulta em

aproximadamente 4,88 mV. Ou seja, cada nivel lógico corresponde a uma variação de 4,88 mV.

1.3 Protocolos de Comunicação de Dados

Dependendo do sensor ou módulo que se está usando é possível que a comunicação aconteça seguindo um protocolo. Podemos entender um protocolo como um “idioma” que todos os dispositivos envolvidos na comunicação precisam “compreender” para que a mensagem seja enviada e recebida. Nesta seção, apresentaremos três desses protocolos que são muito utilizados em aplicações IoT: UART, SPI e I2C.

1.3.1 UART - Universal Asynchronous Receiver/Transmitter

A interface UART é muito comum em microcontroladores e outros dispositivos eletrônicos usados para comunicação serial de dados, constituindo-se numa interface de hardware que facilita a comunicação serial assíncrona entre dispositivos. Ela permite a transmissão e recepção de dados de forma assíncrona, o que significa que não é necessária uma sinalização de clock compartilhado entre o transmissor e o receptor para estabelecer a comunicação. Em vez disso, são usadas configurações de velocidade (baud rate), bits de dados, bits de parada e paridade para estabelecer a comunicação e garantir a sincronização entre os dispositivos conectados.

Alguns conceitos-chave relacionados à UART em microcontroladores são:

- **Baud Rate:** A taxa de baud determina a velocidade de transmissão de dados e é expressa em bits por segundo (bps). Ambos os dispositivos que se comunicam devem ser configurados com a mesma taxa de baud para que a comunicação seja eficaz.
- **Bits de Dados:** Este parâmetro define quantos bits são transmitidos para representar um caractere. Valores comuns são 8 bits, 7 bits ou 9 bits por caractere.
- **Bits de Parada:** São usados para indicar o final de um caractere. Os valores típicos são 1 ou 2 bits de parada.

- **Paridade:** A paridade é uma técnica usada para detectar erros na transmissão. Pode ser par, ímpar ou nenhuma (sem paridade).
- **Transmissão e Recepção Assíncrona:** A UART opera de forma assíncrona, o que significa que os dispositivos não precisam estar sincronizados por meio de um sinal de relógio compartilhado. Isso é alcançado adicionando bits de início no início de cada caractere transmitido.

A Figura 3.3 apresenta um sinal digital seguindo o protocolo UART. Perceba que o dado transmitido é 01011010 que corresponde ao número 76. Seguindo a tabela ASCII, esse valor corresponde ao caracter Z.

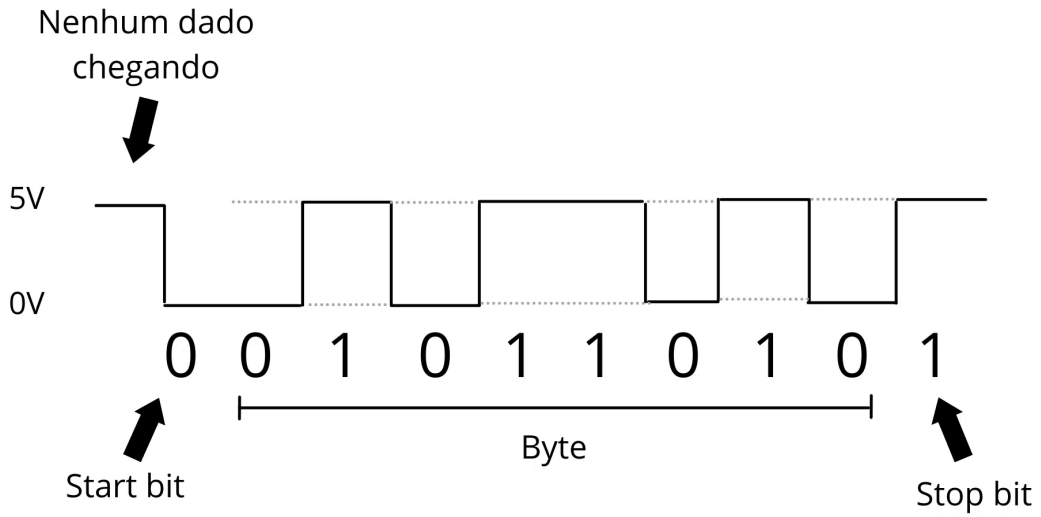


Fig. 3.3 - Representação de um sinal digital seguindo o protocolo UART. (MakerHero)

Saiba Mais! Podemos usar a classe **Serial** para realizar a comunicação UART com o computador em que a placa está conectado e transmitir dados entre eles. Fazemos isso quando queremos usar a função **print** para escrever algum texto em alguma parte do código.

Um porte importante é a ligação entre os dois dispositivos que serão comunicados por meio do protocolo UART. A Figura 3.4 mostra um diagrama

de ligação entre dois dispositivos. Perceba que enquanto um envia o dado (pino Tx) o outro recebe o dado (pino Rx).

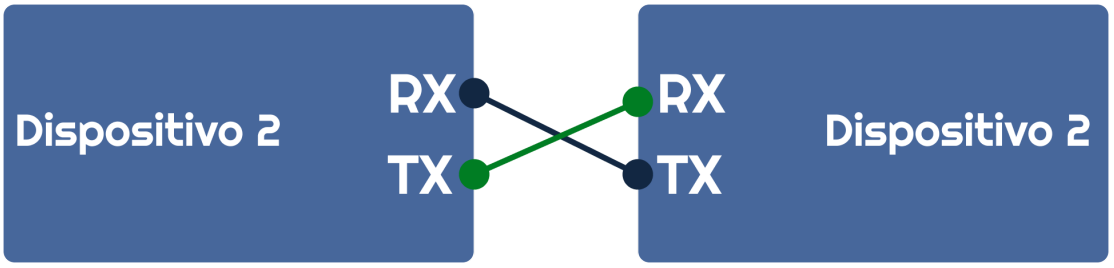


Fig. 3.4 - Diagrama de ligação de dois dispositivos usando o protocolo UART. (Autoria própria)

Abaixo, seguem dois exemplos de configurações da interface UART no ESP8266. No primeiro, são especificados apenas a UART que será usada e a taxa de baud, enquanto na segunda, são definidos os bits de paridade e de parada.

Exemplo 3.1 - Configurando Interface UART no ESP8266 - MicroPython

```
import machine

# Configura a primeira porta UART com uma taxa de baud de 9600 bps
uart = machine.UART(0, baudrate=9600)

# Configura a primeira porta UART com uma taxa de baud de 9600 bps, 8 bits de dados, paridade par e 1 bit de parada.
uart = machine.UART(0, baudrate=9600, bits=8, parity=1, stop=1)
```

1.3.2 SPI - Serial Peripheral Interface

SPI Serial Peripheral Interface é um protocolo de comunicação serial, síncrono, usado para a transmissão de dados entre microcontroladores e periféricos, como memórias flash, sensores, conversores analógico-digital, entre outros. Ele é conhecido por sua simplicidade e eficiência, o que o torna

amplamente utilizado em sistemas embarcados. A seguir, são mostradas suas principais características:

- ➔ **Arquitetura Mestre-Escravo**: O SPI opera em um sistema onde um dispositivo age como mestre e os demais como escravos. O mestre é responsável por iniciar a comunicação, gerar o clock e controlar a seleção do escravo.
- ➔ **Quatro Linhas de comunicação Principais**: O protocolo SPI utiliza principalmente quatro linhas de sinal:
  - ◆ MOSI (Master Out Slave In): Linha de dados que transporta os dados do mestre para o escravo.
  - ◆ MISO (Master In Slave Out): Linha de dados que transporta os dados do escravo para o mestre.
  - ◆ SCK (Serial Clock): Clock gerado pelo mestre para sincronizar a transmissão de dados.
  - ◆ CS/SS (Chip Select ou Slave Select): Usado pelo mestre para selecionar o escravo específico com o qual deseja se comunicar.
- ➔ **Comunicação Full Duplex:** Uma das maiores vantagens do SPI é a capacidade de operar em modo full duplex. Isso significa que os dispositivos podem transmitir e receber dados simultaneamente, já que possui uma linha do mestre para o escravo e outra linha do escravo para o mestre.
- ➔ **Configurações de Modo**: O SPI pode operar em diferentes modos, determinados por duas configurações: polaridade (polarity) e fase (phase). Esses modos determinam a relação entre a transição do sinal de clock e a amostragem de dados.

A comunicação começa quando o mestre seleciona um escravo, colocando o nível lógico baixo na linha CS/SS do escravo (CS/SS = 0V). Em seguida, o mestre gera pulsos de clock na linha SCK para estabelecer o sincronismo da transmissão. Para cada pulso de clock, um bit de dados é transferido entre o mestre e o escravo – o mestre envia um bit para o escravo pela linha MOSI e, ao mesmo tempo, recebe um bit do escravo pela linha MISO.

A Figura 3. 5 apresenta um diagrama que mostra como os dispositivos são conectados. Um ponto importante da comunicação SPI é que para cada novo dispositivo adicionado é necessário um novo pino de “seletor de escravo” (SS) no mestre da comunicação.

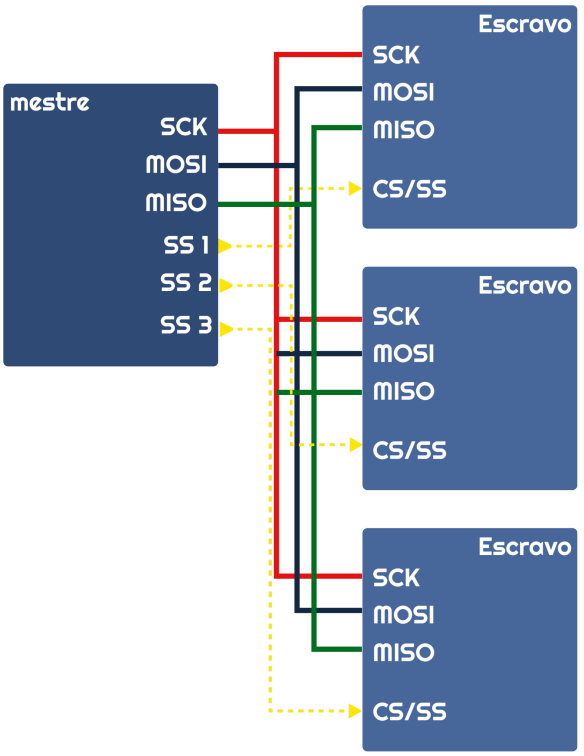


Fig. 3.5 - Diagrama de ligação usando o protocolo SPI. (Autoria Própria)

Abaixo é apresentado um exemplo de utilização do protocolo SPI para controlar um display a partir de um microcontrolador ESP32. São apresentados o código e a montagem do circuito.

Exemplo 3.2 - Exemplo de utilização SPI para conectar um display ao ESP32 - C++

```
#include <Adafruit_ILI9341.h>
#define TFT_DC 2
#define TFT_CS 15
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS,
TFT_DC);
void setup() {
 tft.begin();
 tft.setRotation(1);
 tft.setTextColor(ILI9341_WHITE);
```

```
 tft.setTextSize(2);
 tft.print("Hello IoT");
}

void loop() {}
```

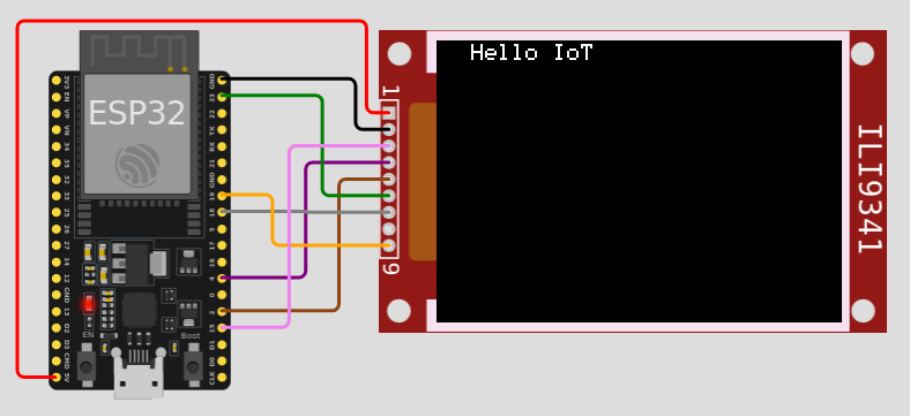


Fig. 6 - Montagem de display `IL9341` interligado ao microcontrolador ESP32, controlado pelo protocolo SPI.

1.3.3 I2C - Inter-Integrated Circuit

O protocolo I2C (Inter-Integrated Circuit) é um protocolo de comunicação bidirecional utilizado para conectar múltiplos dispositivos em uma mesma linha de comunicação. Ele foi desenvolvido pela empresa Philips Semiconductors para permitir que diferentes componentes de um sistema se comuniquem entre si usando um número mínimo de pinos.

Este procolo funciona com a estrutura *master-slave* (mestre-escravo), onde o **master** da comunicação é o dispositivos que controla a comunicação na linha I2C, iniciando e terminando a transmissão de dados e gerando o sinal de clock. Já o **slave** é o dispositivo que responde às requisições do master, enviando ou recebendo dados conforme solicitado.

O número de dispositivos que podem ser conectados a uma linha I2C depende de vários fatores, como o endereçamento dos dispositivos e as características elétricas da linha, como capacitância máxima admitida. Teoricamente, o I2C suporta até 127 dispositivos diferentes em uma mesma

linha, mas na prática, esse número pode ser menor devido as limitações físicas.

O I2C utiliza dois fios para a comunicação: SDA (Serial Data) e SCL (Serial Clock).

- ➔ SCL: É o fio utilizado para o sinal de clock, que sincroniza a transmissão de dados entre os dispositivos.
- ➔ SDA: É o fio utilizado para transmitir os dados entre os dispositivos.

A Figura 3.7 mostra como os dispositivos estão conectados em uma comunicação I2C. Perceba que uma grande vantagem desta comunicação é que, diferente da SPI que precisamos de um pino *slave-select* para cada dispositivo, teremos sempre dois fios para a comunicação de todos os dispositivos, já que o endereçamento é embutido na mensagem do protocolo.

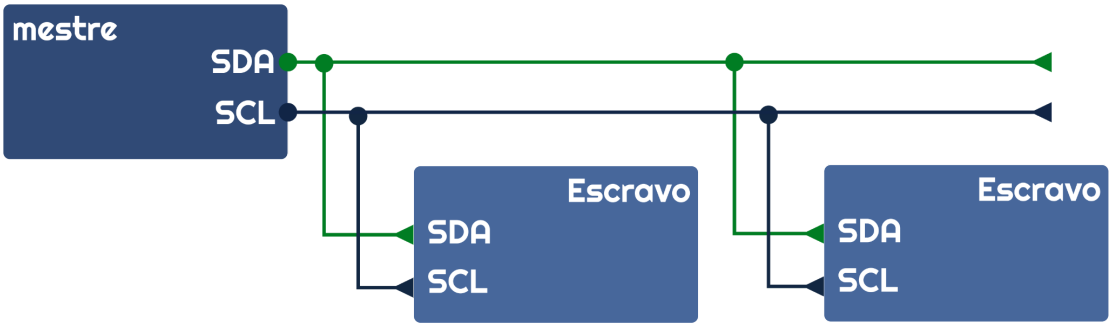


Fig. 3.7 - Diagrama de ligação usando o protocolo SPI. (MakerHero)

O protocolo I2C segue duas regras:

- ➔ **Regra de Escrita -** O master envia um sinal de start, seguido pelo endereço do dispositivo slave com um bit indicando que a operação é de escrita. Em seguida, o master envia os dados, e por fim, envia um sinal de stop.
- ➔ **Regra de Leitura -** O master envia um sinal de start, seguido pelo endereço do dispositivo slave com um bit indicando que a operação é de leitura. O slave então envia os dados, e o master envia um sinal de stop para finalizar a transmissão.

Abaixo é apresentado um exemplo de utilização do protocolo I2C para controlar um display a partir de um microcontrolador ESP32. São apresentados o código e a montagem do circuito.

Exemplo 3.3 - Exemplo de utilização I2C para conectar display ao ESP32 - MicroPython

```
from machine import Pin, I2C
import ssd1306
# ESP32 Pin assignment
i2c = I2C(0, scl=Pin(22), sda=Pin(21))
oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height,
i2c)
oled.text('Hello IoT', 10, 10)
oled.show()
```

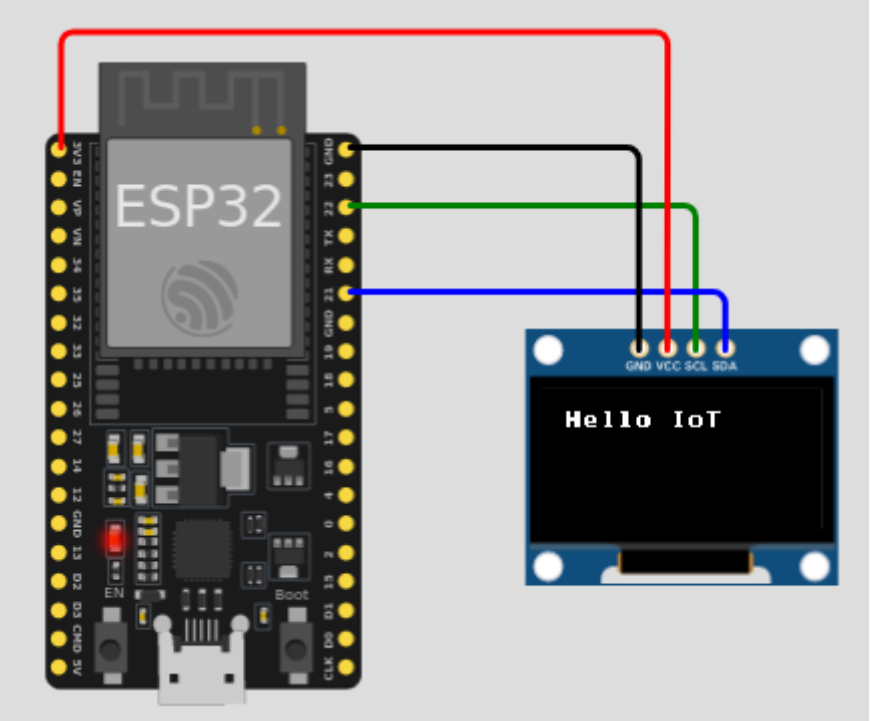


Fig. 8 - Montagem de display `ssd1306` interligado ao microcontrolador ESP32, controlado pelo protocolo I2C.

Exercícios de Fixação

1. Qual a diferença entre um sinal analógico e um sinal digital? Dê exemplos de cada um.
2. Explique de forma sucinta os protocolos de comunicação de dados estudados neste capítulo.
3. No wokwi, crie um projeto que conecta um potenciômetro ao pino ADC de um ESP32 ou Pi Pico W e acione um LED caso o valor esteja acima de um valor máximo especificado.
4. No projeto do item 3, adicione um módulo oled usando protocolo I2C e apresente na tela o valor lido do potenciômetro.

Referências

A. G. D. S. Junior, L. M. G. Gonçalves, G. A. De Paula Caurin, G. T. B. Tamanaka, A. C. Hernandes and R. V. Aroca, "**BIPES: Block Based Integrated Platform for Embedded Systems**," in IEEE Access, vol. 8, pp. 197955-197968, 2020, doi: 10.1109/ACCESS.2020.3035083.
Full text: https://ieeexplore.ieee.org/document/9246562

SANTOS, Bruno P.; SILVA, Luiz A. M.; CELES, Carla S. F. S.; BORGES NETO, João B.; PERES, Bruno S.; VIEIRA, Marcelo A. M.; VIEIRA, Luiz F. M.; GOUSSEVSKAIA, Olga N.; LOUREIRO, Antônio A. F. (2016). **Internet das Coisas: da Teoria à Prática**. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS [SBRC], 31., 30 de maio de 2016, Salvador. Anais [...]. Salvador: UFMG, 2016. p. 1-50.

ASHTON, Kevin. **Entrevista exclusiva com o criador do termo "Internet das Coisas"**. Finep, 2015. Disponível em: <http://finep.gov.br/noticias/todas-noticias/4446-kevin-ashton-entrevista-exclusiva-com-o-criador-do-termo-internet-das-coisas>. Acesso em: 26 de outubro de 2023.

TOTVS. **Aplicações da Internet das Coisas**. Blog Totvs, 2023. Disponível em: <https://www.totvs.com/blog/inovacoes/aplicacoes-da-internet-das-coisas/>. Acesso em: 8 de novembro de 2023.

Usemobile. **IoT: 9 exemplos de aplicativos bem-sucedidos**. Usemobile, 2023. Disponível em: <https://usemobile.com.br/iot-9-exemplos-de-aplicativos/>. Acesso em: 8 de novembro de 2023.

Google. **Breaking down language barriers with augmented reality**. YouTube, 2023. Disponível em: <https://www.youtube.com/watch?v=lj0bFX9HXeE>. Acesso em: 8 de novembro de 2023.

ZUP. **A arquitetura da Internet das Coisas**. 30 de agosto de 2023. Disponível em: https://www.zup.com.br/blog/a-arquitetura-da-internet-das-coisas. Acesso em: 28 de novembro de 2023.

AMAZON WEB SERVICES. **O que é MQTT?** 20 de julho de 2023. Disponível em: https://aws.amazon.com/pt/what-is/mqtt/. Acesso em: 28 de novembro de 2023.

GTA/UFRJ. **COAP: Protocolo de comunicação para a Internet das Coisas**. 8 de março de 2019. Disponível em: https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/coap/. Acesso em: 28 de novembro de 2023.

HIVEMQ. **MQTT vs HTTP: Protocolos para IoT e IIoT**. 21 de junho de 2023. Disponível em: https://www.hivemq.com/article/mqtt-vs-http-protocols-in-iot-iiot/. Acesso em: 28 de novembro de 2023.

EMBARCADOS. **AMQP: Protocolo de comunicação para IoT**. 10 de maio de 2023. Disponível em: https://embarcados.com.br/amqp-protocolo-de-comunicacao-para-iot/. Acesso em: 28 de novembro de 2023.

BOM CURSO!